Final Technical Report to the

Office of Naval Research

by

R. V. Helgason, J. L. Kennington, and K. H. Lewis
Department of Computer Science and Engineering
SOUTHERN METHODIST UNIVERSITY
School of Engineering and Applied Science
Dallas, Texas 75275–0122

for

Shortest Path Algorithms on Grid Graphs with Applications to Strike Planning

ONR Contract Number N00014–96–1–0315

SMU Contract Number 5–25182

12 February 1997

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	unrestricted
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION	6b. OFFICE SYMBOL (*If applicable*)	7a. NAME OF MONITORING ORGANIZATION	
Southern Methodist Univ.	CSE	Office of Naval Research	
6c. ADDRESS (*City, State, and ZIP Code*)		7b. ADDRESS (*City, State, and ZIP Code*)	
6425 Airline Drive Dallas TX 75275-0122		800 North Quincy Street Arlington VA 22217-5660	
8a. NAME OF FUNDING / SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
ONR			
8c. ADDRESS (*City, State, and ZIP Code*)		10. SOURCE OF FUNDING NUMBERS	
800 North Quincy Street Arlington VA 22217-5660		PROGRAM ELEMENT NO.	PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)

12. PERSONAL AUTHOR(S)
R. Helgason, J. Kennington, and K. Lewis

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
technical	FROM 2/1/96 TO 1/31/97	97 - 2 - 12	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	cruise missile mission planning

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

This document contains three technical reports related to the problem of developing mission plans for a cruise missile. The first report presents algorithms for finding the safest path through a graph and for finding the safest corridor in a grid graph. The second report presents anew effective algorithm for finding a path from a designated origin to a designated destination which avoids all threat regions represented by circles. This algorithm does not require the imposition of a grid over the region and avoids threats by moving along the edges of triangles which enclose the threats. The third report gives an algorithm which obtains short paths which satisfy some side restriction on the probability of success. These paths may take short cuts through the threat regions as long as the side constraint is satisfied.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Jeffery L. Kennington	22b. TELEPHONE (*Include Area Code*) (214) 768-3088 / 22c. OFFICE SYMBOL CSE

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Table of Contents

I. Statement of Work

Generally a missile is called a cruise missile if its speed is subsonic, if it uses a built–in global positioning navigation system (GPS/INS), and if its range is at least several hundred miles. A specific mission for a cruise missile is programmed by specifying a sequence of waypoints and including any TERCOM maps that are available. The missile uses its on–board computational facilities and its GPS/INS system to guide it to its target.

During the 1990s, cruise missiles launched from ships and submarines were used with great success in the Persian Gulf. Targets which may be hundreds of miles inland have been successfully attacked by these Navy weapons stationed in relatively safe positions offshore. In addition, cruise missiles have been used to disable SAM sites, making manned aircraft strikes safer.

By representing threat regions as circles, the problem of developing a single mission for a single cruise missile can be viewed as a computational geometry problem in which we seek a path composed of line segments from a launch site to a target site which skirts selected threat regions. Using current technology, missions are developed using a two–step process. In the first step, a mission planner uses a two–dimensional map to manually select a path from a launch site to the target. In the second step a software system includes the vertical dimension and develops an estimate of the probability of a successful mission. This process may be repeated several times until the mission planner is satisfied with the mission plan. In this research, we present three algorithms designed to help automate the first step of the mission planning process.

II. Finding Safe Paths

In this study, we lay a grid over the military theatre of interest and create a grid graph [N, A]. The points s ε N and t ε N are defined to be the grid points nearest the missile and target, respectively. For each edge we define a traversal probability. Traversal probabilities within a threat region are set to values which are proportional to the distance from the midpoint of an edge to the center of the threat. Edges which are outside the threat regions have traversal probabilities very close to 1. By applying a slight modification of Dijkstra's algorithm, we can easily find the safest path from s to t. The set of nodes which appear in near−optimal safe paths has been called the safe corridor. This manuscript (which appears in Appendix A) presents a proof of correctness for both the problem of finding the safest path and the problem of finding a safe corridor in a grid graph.

III. Grid–Free Algorithms

In this study, we present a new simple, yet surprisingly effective algorithm that automatically finds a path from the launch site to the target site which avoids all threats and does not require the use of a grid. The circumscribed triangle algorithm constructs a mission by moving from a given waypoint tangentially to the circumference of nearby threat regions. These directions are determined by constructing a circumscribed triangle around a threat region. This basic strategy is embedded within an efficient branch–and–bound framework. On twenty randomly generated test problems, this algorithm worked extremely well in all cases. This study is documented in Appendix B.

IV. Mission Planning With A Probability Side Constraint

After consultation with analysts at the Naval Surface Warfare Center at Dahlgren, Virginia, we discovered that real mission plans frequently penetrate enemy threat regions. In this investigation, we extended the circumscribed triangle algorithm to incorporate a side constraint on the probability of a successful traversal. This allows the missile to take short cuts through threat regions, as long as the probability of success meets some user−specified criteria. This work is documented in Appendix C.

Appendix A

Finding Safe Paths in Networks

Technical Report 96-CSE-7

Finding Safe Paths in Networks

by

R. V. Helgason
(helgason@seas.smu.edu)

J. L. Kennington
(jlk@seas.smu.edu)

K. H. Lewis
(klewis@seas.smu.edu)

Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275-0122

revised September 1996

Abstract

The problem of constructing strike plans for cruise missiles motivated the formulation of two new graph problems: the safest path problem and the safe corridor problem. The idea is to construct a flight path for a cruise missile through a military theatre which results in a high probability of mission success. Exact algorithms for both problems are presented along with a proof of correctness. The notion of a safe corridor is illustrated on several randomly generated strike planning problems involving a single cruise missile and a single target protected by several surface-to-air missile sites. The network is defined by a grid graph over the terrain of interest.

Acknowledgement

This research was supported in part by the Office of Naval Research under contract number N00014-96-1-0315.

1 Introduction

Let $G = [N, A]$ be a directed graph consisting of a set N of nodes and a set A of arcs whose elements are ordered pairs of distinct nodes. A *directed path* in G from node s to node t is an alternating sequence of nodes and arcs from G of the form $P_{st} = \{s = i_1, (i_1, i_2), i_2, (i_2, i_3), i_3, ..., i_{k-1}, (i_{k-1}, i_k), i_k = t\}$. Let p_{ij} denote the probability of a safe traversal over the arc (i,j). Then the probability of a safe traversal over the directed path P_{st} is given by $p_{i_1 i_2} p_{i_2 i_3} \cdots p_{i_{k-1} i_k}$. Given $s \in N$ and $t \in N$, the *safest path problem* is to find a directed path in G having the largest probability of safe traversal. Given $R \leq 1$, $s \in N$ and $t \in N$, the *safe corridor problem* is to find the set of nodes $C \subseteq N$ such that for each $n \in C$ there exists a path containing n whose traversal probability is at least Rp^*, where p^* is the traversal probability of the safest path in G.

The objective of this investigation is to develop an efficient algorithm for the safest corridor problem and demonstrate how this can be used to plan flight paths for cruise missiles. The safe corridors generated allow a mission planner to develop flight paths which maximize mission success.

The safest path problem is closely related to the classical shortest path

problem which is discussed in numerous books (see for example [1, 2, 3, 7, 10, 11, 12, 13, 14, 18]). Excellent surveys of the many shortest path problem variations may be found in [5, 8, 9]. The idea of a safe corridor was first introduced to the authors by Solka and Rogers [15], who were developing a safe corridor using Dijkstra's shortest path algorithm. Parallel versions of their algorithms may be found in [16, 17].

2 The Safest Path Algorithm

The safest path algorithm presented in this section is a straight-forward adaptation of Dijkstra's classical algorithm [6] for the one-to-one shortest path problem. The input is a directed graph $[N, A]$ with node set $N = \{1, 2, ..., n\}$ and arc set A. Associated with each arc (i, j) is the probability of a successful traversal, p_{ij}. A safest path is desired from the origin $s \in N$ to the destination $t \in N$, and it is assumed that a path from s to t exists.

2.1 The Algorithm

The safest path algorithm begins at node s and constructs a safest path tree T in which the safest path from s to any node in T is known. When node t

is placed in the tree we have a safest path from s to t and the algorithm terminates. The algorithm uses two working arrays and working sets S_k which contain the nodes in the current T at iteration k. The array d_j denotes the probability of traversal from s to j and the array b_j denotes the predecessor of node j, i.e. if $(i,j) \in T$, then $b_j = i$. At each iteration, T is enlarged by one node, i.e. $|S_k| = k$. Hence, in at most $|N|$ iterations the algorithm will terminate.

Algorithm : safest path

Inputs : N, A, s, t, p_{ij}

Outputs : T, b_j for all nodes $j \in T$, and d_t

Assumption : There exists a path from s to t in $[N, A]$.

begin

$S_0 \leftarrow \emptyset$; $\quad k \leftarrow 0$;

$d_j \leftarrow -1$ for all $j \in N$;

$d_s \leftarrow 1$; $\quad b_s \leftarrow 0$;

while $t \notin S_k$ **do**

begin

select $i_k \in N \backslash S_k$ **such that** $\forall j \in N \backslash S_k$, $d_{i_k} \geq d_j$;

$S_{k+1} \leftarrow S_k \cup \{i_k\}$; $\quad k \leftarrow k+1$;

for all $j \in N \backslash S_k$ such that $(i_k, j) \in A$ do

begin

if $d_j < d_{i_k} p_{i_k j}$ then $\quad d_j \leftarrow d_{i_k} p_{i_k j}$; $\quad b_j \leftarrow i_k$;

end;

end;

end;

An algorithm for the one-to-all safest path problem can be obtained by modifying the termination criterion to "while $N \backslash S_k \neq \emptyset$ do."

2.2 Correctness of the Algorithm

In this section we show that the safest path algorithm terminates with the correct solution. The proof is based on induction and relies on the fact that the safest path from s to every node in the safest path tree is known.

Theorem. When the safest path algorithm terminates, d_t is the probability of the safest path from s to t in $[N, A]$.

Proof. We proceed by finite induction using the predicate

$P(m)$: (a) for all $i \in S_m$, d_i is optimal with an optimal path from s to i lying in S_m, and

(b) for all $j \in N \backslash S_m$ with $d_j \geq 0$, d_j is the largest traversal probability for a path from s to j in which all nodes except j lie in S_m, and such a path can be decomposed into a subpath from s to b_j and the subpath consisting of b_j followed immediately by j.

When $S_0 = \emptyset$, only $d_s > 0$. So $S_1 = \{s\}$ and $d_s = 1$. Since any path with probability 1 must be optimal, d_s is optimal and the degenerate optimal path from s to s lies in S_1. Thus (a) holds. Also, the only path from s to a node j in $N \backslash S_1$ consists of s and the arc connecting s to j. For any such point j, d_j was set to $d_s p_{sj} = p_{sj}$, the traversal probability associated with the only path from the single node s of S_1 to j, using only $b_j = s$ and j. So (b) holds. Thus $P(1)$ is true.

Let k be an arbitrary fixed integer such that $1 \leq k < n$. Assume that $P(1), \ldots, P(k)$ are all true. Consider S_{k+1} and $i_k \in S_{k+1} \backslash S_k$. Since $P(k)$ is true, d_i is optimal for all $i \in S_k$, and optimal paths from s to those nodes also lie in S_k. Let P_0 be the path from s to i_k, with traversal probability p_0,

consisting of the path from s to b_{i_k} followed immediately by i_k.

Suppose (a) does not hold for S_{k+1}. It must be that there is a path P_1 from s to i_k with traversal probability $p_1 > d_{i_k}$. Furthermore, this path must contain at least one node of $N \backslash S_{k+1}$. Let q be the first node of $N \backslash S_{k+1}$ on path P_1, so that P_1 decomposes into two subpaths P_2 from s to q and P_3 from q to i_k. Let p_2 and p_3 be the traversal probabilities of paths P_2 and P_3, respectively. All nodes of P_2 except q lie in S_k. So by (b), d_q is the largest traversal probability for a path from s to q in which all nodes except q lie in S_k. But when i_k was selected, $d_{i_k} \geq d_q$. So $p_1 = p_2 p_3 \leq d_q p_3 \leq d_{i_k} p_3 \leq d_{i_k}$, a contradiction. Thus (a) holds for S_{k+1}.

We now show (b) holds for S_{k+1}. Let $r \in N \backslash S_{k+1}$ with $d_r \geq 0$. Let P_4 be the path with the largest traversal probability for a path from s to r in which all nodes except r lie in S_{k+1} and let that probability be p_4. Let u be the last node of S_{k+1} on P_4.

Case 1. Suppose that before i_k was selected, $d_r = -1$. Then after i_k was selected, the only paths from s to r with nodes except for r lying entirely in S_{k+1} have i_k and r as the last two nodes and (b) holds for r.

Case 2. Suppose that before i_k was selected, $d_r \geq 0$. Thus before i_k was selected, since (b) holds for S_k, d_r was the largest traversal probability for a

path from s to r in which all nodes except r lie in S_k. Let p_r be the value of d_r at that point.

Subcase 2.1. Assume u is i_k. Then $p_4 = d_{i_k} p_{i_k r}$. Thus following the adjustment to d_r after i_k was selected $d_r = p_4$ since p_4 is the largest probability path and (b) holds for S_{k+1}.

Subcase 2.2. Assume u is not i_k, but is some other node $j \in S_{k+1}$. Thus P_4 will not contain i_k and no adjustment was made to d_r after i_k was selected, so that $d_r = p_r$ and (b) holds for S_{k+1}.

∎

3 The Safe Corridor Algorithm

For applications in the area of strike planning, there are many alternate optima. In addition, there are many paths that come very close to being optimal. The set of nodes which appear in near-optimal safe paths is called the *safe corridor.*

The safe corridor algorithm constructs two safest path trees, T_1 and T_2. T_1 records the safest path from s to any node in its tree, and T_2 records the safest path from t to any node in its tree. All nodes in the graph are

initially assigned a negative probability of successful arrival from the source and from the destination. Some nodes may not be updated with non-negative probabilities, but all nodes near a safest path will be updated.

For any node j, let δ_j denote the greatest probability of successful traversal from node j to the destination, t, let β_j denote the predecessor of node j in T_2, and let ζ_k denote the set containing the current nodes in T_2 at iteration k. The greatest probability of successful traversal from the source to the destination via node j is $d_j\delta_j$, where d_j and δ_j are the safest route probabilities from s to j and from j to t, respectively, and have been updated with nonnegative probabilities. For any node i along the safest path from s to t, $d_i\delta_i = d_t$. Also $\delta_s = d_t$, since $p_{ij} = p_{ji}$. If $d_j\delta_j/d_t$ is very close to 1, then node j is not on a safest path, but the probability of successful traversal from s to t via j is almost as great as that of a safest path. Hence, a safe corridor can be generated by including all nodes j for which $d_j\delta_j$ is within a given range near d_t.

The safe corridor $C \subseteq N$ is determined by a parameter $R \leq 1$. If $R = 1$, then every node in C is on some optimal path from s to t. If $R = 0.9$, then the nodes in C are on paths whose traversal probability is at most 10% less than the probability of the safest path. If $R = 0$, then $C = N$. The algorithm

is given as follows:

Algorithm : safe corridor

Inputs : N, A, s, t, p_{ij}, R

Outputs : C

begin

$S_0 \leftarrow \emptyset$; $k \leftarrow 0$;

$d_j \leftarrow -1$, $\delta_j \leftarrow -1$ for all $j \in N$;

$d_s \leftarrow 1$; $b_s \leftarrow 0$; $\delta_t \leftarrow 1$; $\beta_t \leftarrow 0$;

while $t \notin S_k$ **do**

begin

select $i_k \in N \backslash S_k$ **such that** $\forall j \in N \backslash S_k$, $d_{i_k} \geq d_j$;

$S_{k+1} \leftarrow S_k \cup \{i_k\}$; $k \leftarrow k+1$;

for all $j \in N \backslash S_k$ **such that** $(i_k, j) \in A$ **do**

begin

if $d_j < d_{i_k} p_{i_k j}$ **then** $d_j \leftarrow d_{i_k} p_{i_k j}$; $b_j \leftarrow i_k$;

end;

end;

$\zeta_0 \leftarrow \emptyset$; $k \leftarrow 0$;

while $s \notin \zeta_k$ do

begin

select $h_k \in N \backslash \zeta_k$ such that $\forall j \in N \backslash \zeta_k$, $\delta_{h_k} \geq \delta_j$;

$\zeta_{k+1} \leftarrow \zeta_k \cup \{h_k\}$; $\quad k \leftarrow k+1$;

for all $j \in N \backslash \zeta_k$ such that $(j, h_k) \in A$ do

begin

if $\delta_j < \delta_{h_k} p_{jh_k}$ then $\delta_j \leftarrow \delta_{h_k} p_{jh_k}$; $\quad \beta_j \leftarrow h_k$;

end;

end;

$C \leftarrow \emptyset$;

for all $j \in N$

begin

if $d_j \delta_j \geq R(d_t)$ then $C \leftarrow C \cup \{j\}$;

end;

end;

The value of R must be carefully chosen for each problem. A value too close to 1 can exclude valuable options. However, as the value of R is reduced, the safe corridor can very quickly expand to include a large portion of the

grid.

4 The Two-Dimensional Strike Planning Problem on a Grid Graph

Strike planning for cruise missiles involves finding a flight path through a combat theatre that involves threats. The most serious threat for a cruise missile is an air defense system consisting of a search radar, a fire-control radar, and a SAM weapon (see [19]). The effectiveness of the SAM system depends on the total amount of time it has to engage and destroy the cruise missile. The search radar is usually always active, while the fire-control radar only becomes active when needed. The search radar spends some time in determining if a missile detect is real or is a false alarm. A cruise missile flies at low altitude in an attempt to avoid detection. The fire-control radar is turned on only after search radar personnel decide that they have a real target. There is some warmup required and some time needed to aim and fire the SAM. The less time a cruise missile spends near one of these sites, the higher the probability of mission success.

For this application, we lay a grid over the area and form a grid graph $[N, A]$. The points $s \in N$ and $t \in N$ are the grid points nearest the missile and target, respectively. The traversal probabilities are related to the location of the threats. Given $R < 1$, the objective is to determine $C \subseteq N$, the safe corridor through $[N, A]$.

To demonstrate the safe corridor algorithm, we developed a random problem generator. Each interior point in the grid has degree 8 as illustrated in Figure 1. The missile is randomly placed on the left vertical boundary and the target is randomly placed on the right vertical boundary. The SAM systems are placed at random nodes in $[N, A]$. Three test problems and the corresponding safe corridors are illustrated in Figures 2, 3, and 4. The traversal probabilities are in the interval [0.1, 0.99]. Probabilities within a threat region (determined by the distance to the fire-control radar) are proportional to the distance from the midpoint of an edge to the fire-control unit. Horizontal and vertical edges which are outside threat regions have traversal probabilities of 0.99. Diagonal edges have traversal probabilities of 0.985. Hence, traversing a diagonal arc is slightly safer than traversing one horizontal and one vertical arc connecting the endpoints of the diagonal arc.

Figures 1 through 4 about here

5 Summary and Extensions

This manuscript presents a new graph problem which we call the safest path problem. A Dijkstra-like algorithm for this problem is presented along with a correctness proof. This work was motivated by the problem of developing strike plans and we demonstrate how our algorithm can be used to develop safe corridors on two-dimensional grid graphs.

The work presented in this manuscript can be extended to three dimensions. Even though the algorithm is very fast, problem size could eventually lead to excessive computation time. It is also the case that repeated flights over a threat may cause threat personnel to become more alert so that the second flight near a threat may have a lower probability of successful traversal than the first. Planning for multiple strikes would need to account for this. A description of this problem along with solution approaches may be found in [4, 20].

References

[1] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, NJ 1993.

[2] C. Berge and A. Ghouila-Houri, *Programming, Games, and Transportation Networks*, John Wiley and Sons, New York, NY 1962.

[3] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, Englewood Cliffs, NJ 1987.

[4] A. Boroujerdi, C. Dong, Q. Ma, and B. Moret, "Joint Routing in Networks," undated technical report, Department of Computer Science, University of New Mexico, Albuquerque, NM.

[5] N. Deo and C. Pang, "Shortest Path Algorithms: Taxonomy and Annotation," *Networks* 14 (1984) 275 - 323.

[6] E. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik* 1 (1959) 269 - 271.

[7] S. Even, *Graph Algorithms*, Computer Science Press, Rockville, MD 1979.

[8] G. Gallo and S. Pallottino, "Shortest Path Methods: A Unifying Approach," *Mathematical Programming Study* 26, (1986) 38 - 64.

[9] G. Gallo and S. Pallottino, "Shortest Path Algorithms," *Annals of Operations Research* 13, (1988) 3 - 79.

[10] T. Hu, *Combinatorial Algorithms*, Addison-Wesley, Reading, MA 1982.

[11] P. Jensen and J. Barnes, *Network Flow Programming*, John Wiley and Sons, New York, NY 1980.

[12] E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Reinhart, and Winston, New York, NY 1987.

[13] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ 1987.

[14] R. Rockafellar, *Network Flows and Monotropic Optimization*, John Wiley and Sons, New York, NY 1984.

[15] J. Solka and G. Rogers, Strike Planning Briefing at the Dahlgren Division of the Naval Surface Warfare Center, Dahlgren, VA Spring 1995.

[16] J. Solka, J. Perry, B. Poellinger, and G. Rogers, "Fast Computation of Optimal Paths Using a Parallel Dijkstra Algorithm with Embedded Constraints," *Neurocomputing* 8, (1995) 195 - 212.

[17] J. Solka, J. Perry, B. Poellinger, and G. Rogers, "Autorouting Using a Parallel Dijkstra Algorithm with Embedded Constraints," undated technical report, The Naval Surface Warfare Center, Dahlgren, VA.

[18] R. Tarjan, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadephia, PA 1983.

[19] M. Zuniga and P. Gorman, "Threat Site Overflight Modeling for Strike Route Optimization," undated technical report, Naval Research Laboratory, Washington, D.C.

[20] M. Zuniga, J. Uhlmann, and J. Hofmann, "The Interdependent Joint Routing Problem: Description and Algorithmic Approach," undated technical report, Naval Research Laboratory, Washington, D.C.

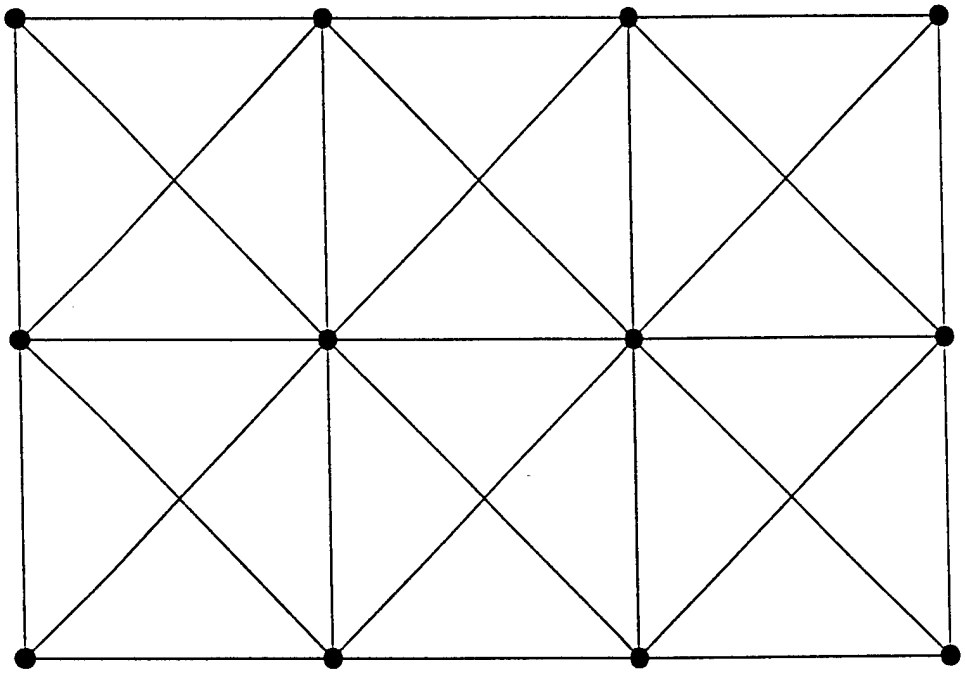

Figure 1. A 3 x 4 Grid Graph

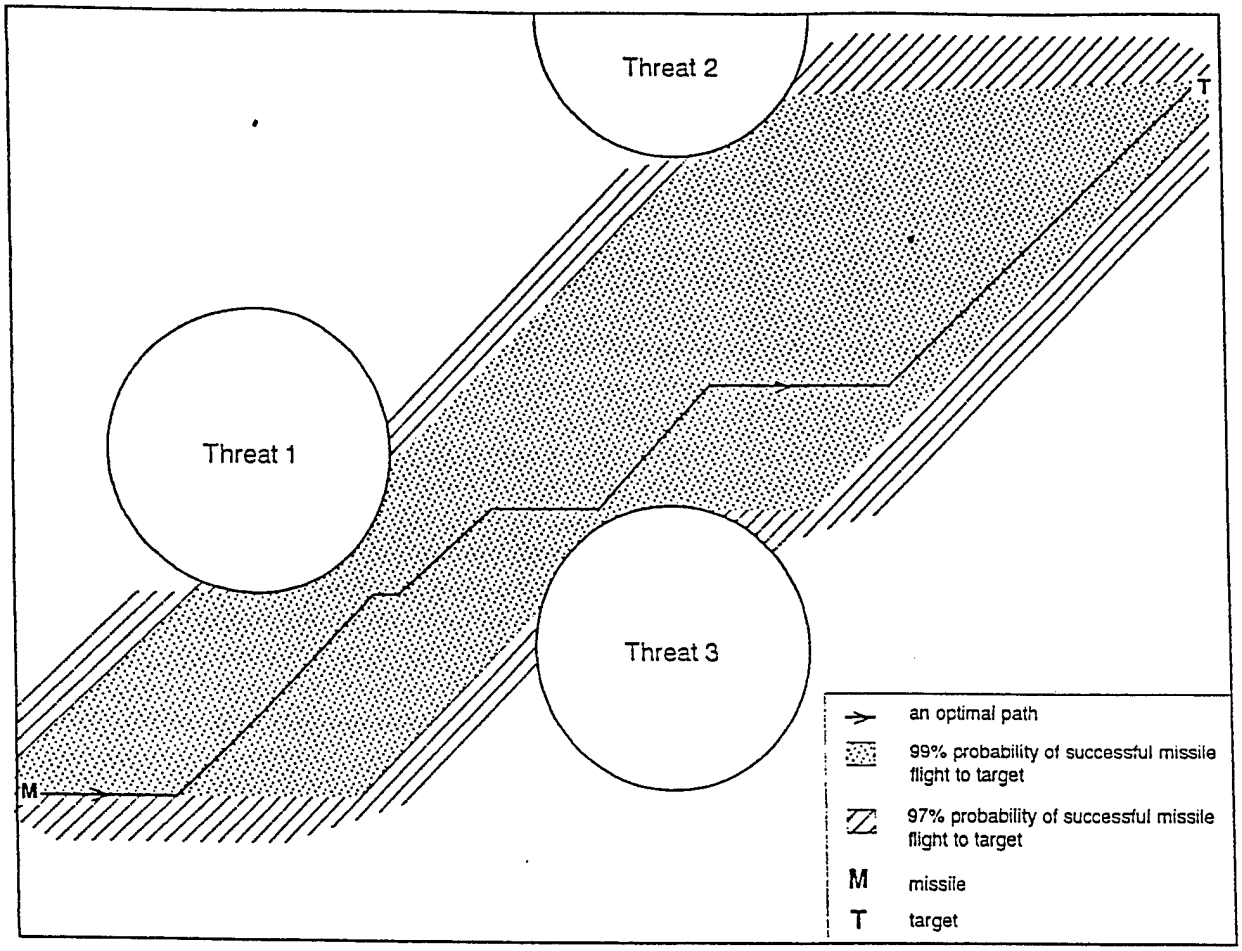

Figure 2. Illustration of Safe Corridor Between Missile and Target

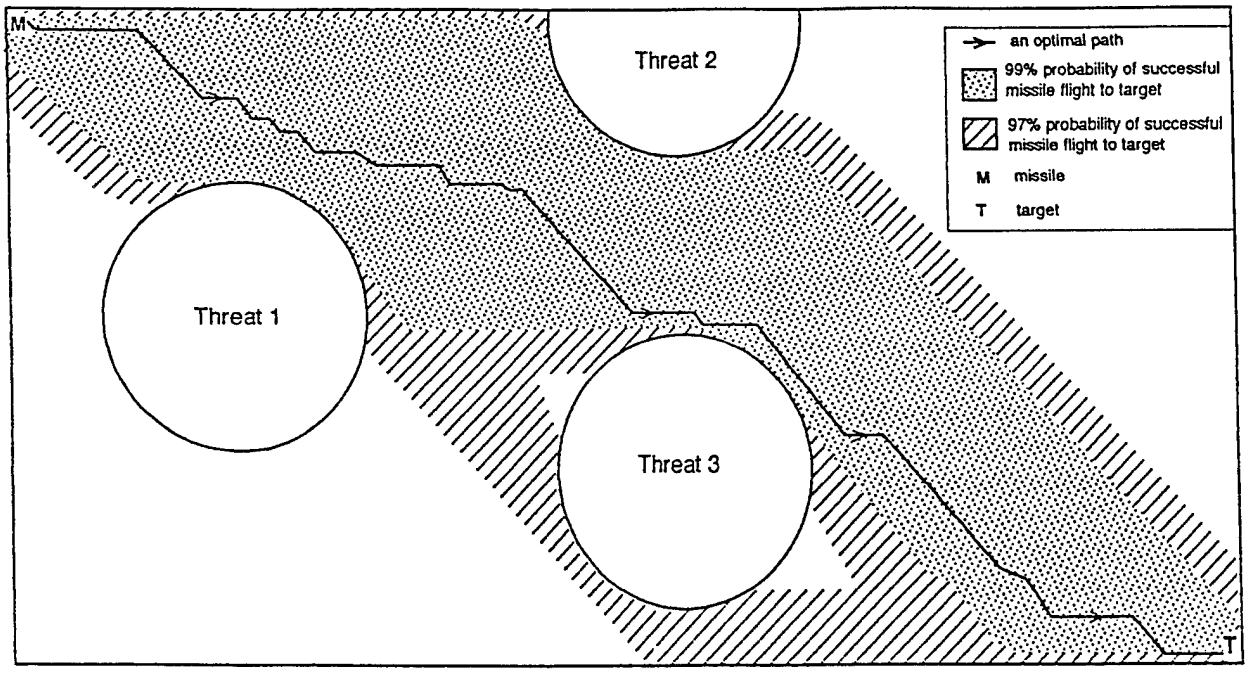

Figure 3. Illustration of Safe Corridor

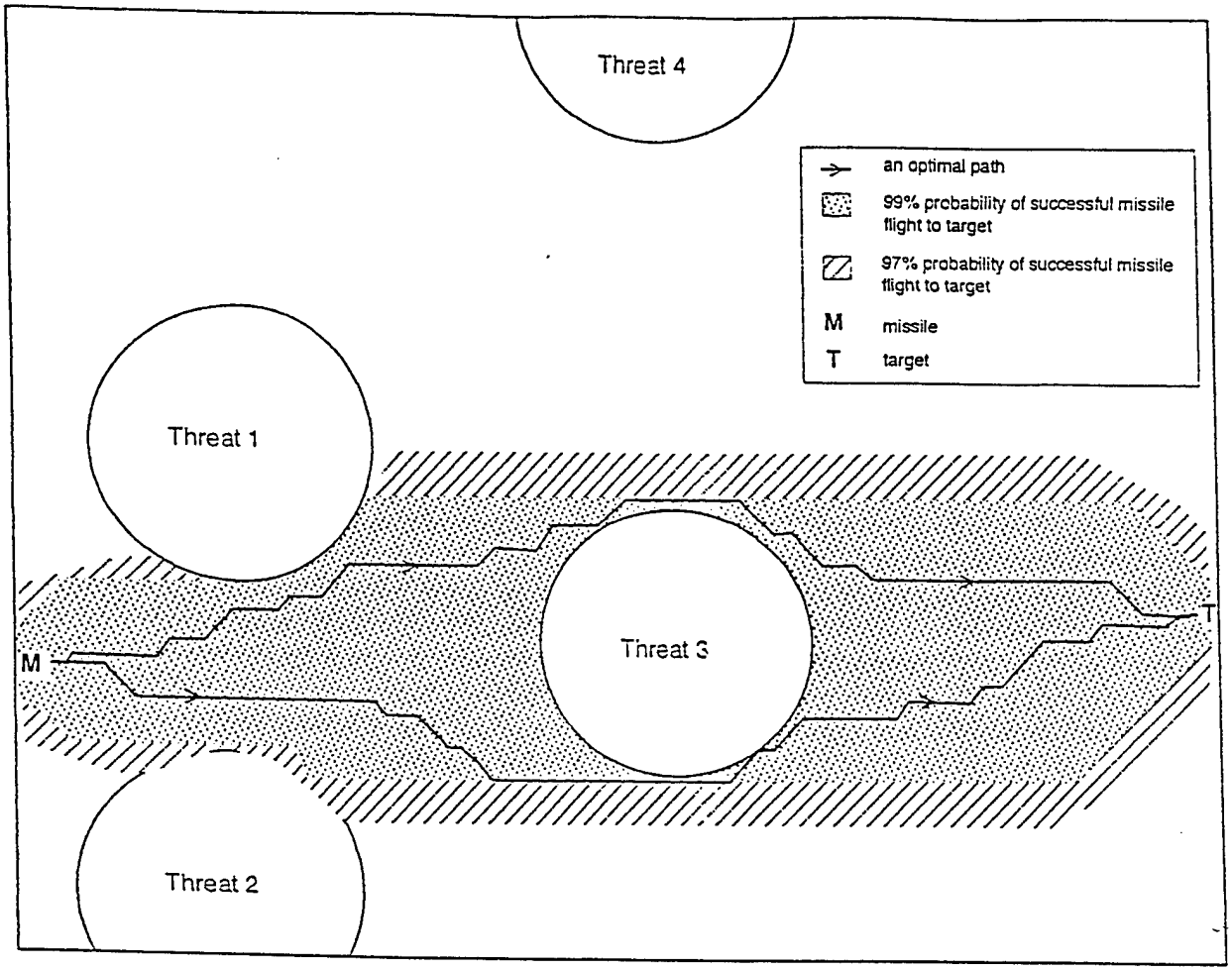

Figure 4. Illustration of Alternate Optima on Either Side of Threat

Appendix B

Grid–Free Algorithms for Mission Planning

Technical Report 96-CSE-10

Grid-Free Algorithms for the Two-Dimensional Mission Planning Problem for Cruise Missiles

by

R. V. Helgason
(helgason@seas.smu.edu)

A. C. Jayasuriya
(acj@seas.smu.edu)

J. L. Kennington
(jlk@seas.smu.edu)

K. H. Lewis
(klewis@seas.smu.edu)

Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275-0122

September 1996

Abstract

The problem of mission planning for a cruise missile in two dimensions can be viewed as a mathematical problem in the area of computational geometry in which we seek a path composed of a set of line segments from a given missile location to a given target location which does not pass through any SAM radar site. The shortest path requiring the fewest number of turns is most desirable. Two algorithms for this problem have been developed and compared in an empirical study.

Acknowledgement

This research was supported in part by the Office of Naval Research under contract number N00014-96-1-0315.

1 INTRODUCTION

For this investigation, we classify a missile as a *cruise missile* if its speed is sub-sonic, it uses a built-in global positioning navigation system (GPS/INS), and its range is at least 300 miles. A specific mission for a cruise missile is programmed by simply specifying a sequence of coordinates. The missile uses its on-board computational facilities and its GPS/INS system to guide it through this sequence of points. At present, once a cruise missile is launched, its mission cannot be modified.

A cruise missile can be destroyed in flight by a SAM air defense system consisting of three components: a search radar, a fire-control radar, and a SAM weapom. A complete description of this system may be found in Zuniga and Gorman [7]. According to Zuniga and Gorman, the effectiveness of the SAM system depends on the total amount of time it has to engage and destroy the cruise missile. The search radar only becomes active when needed. Some time is spent to determine if a missile detect is real or false. If the detect is determined to be real, then the fire-control radar is turned on to aim and fire the SAM. The search radar has the same effective range which is represented by a circle in two dimensions.

Mission planning for a single cruise missile involves constructing a flight path from a given origin to a given destination through a combat theatre having SAM sites. In two dimensions, the SAM sites are represented as circles and the objective is to find a set of line segments linking the origin to the destination which do not pass through any SAM threat circle.

Most research investigations in the area of mission planning begin by placing a grid over the combat theatre and applying a modification of Dijkstra's algorithm (see [2]) to obtain a safe path from the origin to the destination (see [1, 3, 4, 5, 7, 8]). There are two disadvantages to this approach. The grid graph can become very large, particularly for the three-dimensional case, and consequently the algorithms can be very time consuming. Also, the resulting paths may involve too many line segments. A path with only a few line segments is viewed as better than one with many line segments. The objective of this investigation is to present new algorithms for mission planning that do not require the use of a grid.

2 THE MISSION PLANNING PROBLEM

The problem of finding a *feasible mission plan* can be defined mathematically as follows:

Given two points M and $T \in R^2$ and K circles each with center $X_1, ..., X_K \in R^2$ and radii $r_1, ..., r_K$; find a set of line segments $[Y_1, Y_2]$, $[Y_2, Y_3]$, ..., $[Y_S, Y_{S+1}]$ with $Y_1 = M$ and $Y_{S+1} = T$ such that no line segment intersects any circle.

A feasible mission plan through a theatre having five SAM sites is illustrated in Figure 1. The quality of a mission plan is measured by the length of the path and the number of segments. The best path is the straight line from M to T. If this is not feasible, then we seek a short path with only a few segments.

Figure 1 about here

---

3 MISSION PLANNING ALGORITHMS

In this section, two new algorithms are presented for solving the two-dimensional mission planning problem. The projection algorithm begins with a set of line segments linking M to T. If the segments intersect any of the threats, then by projecting the center of the threat circles onto the nearest line segment, new line segments can be constructed which usually avoid the threats. The circumscribed triangle algorithm creates paths around the threats by creating triangles which enclose the threats. The paths are constructed by linking the edges of these special triangles.

3.1 The Projection Algorithm

The projection algorithm begins with one or more line segments linking M (the missile) to T (the target). If this is feasible, then the algorithm terminates. Otherwise, additional break points are created to guide the missile around the threats. The line segment $[M, T]$ illustrated in Figure 2 has been deflected around the threat by adding a break point between M and T.

Let X_1 denote the center of the threat circle with radius r_1. Let Z be the projection of X_1 onto the line segment $[M, T]$. Let W be a new point ob-

tained by moving from X_1 in the direction $Z - X_1$ to a point slightly beyond the circle. The new path is $[M, W]$ $[W, T]$.

Figure 2 about here

Deflecting a path around a threat can result in the new path passing through a different threat as illustrated in Figure 3. This requires another application of the projection procedure. By continuing in this way, we can in most cases obtain a set of line segments that link M to T and avoid all of the threats.

Figure 3 about here

The basic algorithm is called *projector* and the algorithm is defined as follows:

Algorithm : *projector*

Inputs :

M - missile location

T - target location

K - number of threats

$X_1, ..., X_K$ - center points of the threats

$r_1, ..., r_K$ - radii of the threats

L - number of segments in the initial path

$I_1, ..., I_{L+1}$ - the initial path

Outputs :

S - number of segments in solution

$Y_1, ..., Y_{S+1}$ - the points in the solution path

begin

$Y_s \leftarrow I_s$ for $s = 1, ..., L + 1$;

$s \leftarrow 1$, $k \leftarrow 1$, $S \leftarrow L$;

while $(s \leq S)$ **do**

begin

$Z \leftarrow$ project (X_k, Y_s, Y_{s+1});

if (Z is on the line segment $[Y_s, Y_{s+1}]$) **then**

```
    begin
      if (distance (Z, X_k) ≤ r_k) then
      begin
        W ← X_k + 1.1(r_k)(Z − X_k)/distance (Z, X_k)
        for i = S + 1 to s + 1 do Y_{i+1} ← Y_i;
        Y_{s+1} ← W, S ← S + 1, s ← 1, k ← 0;
      end;
    end;
    k ← k + 1;
    if (k >K) then k ← 1, s ← s + 1;
  end;
end;


procedure distance (U, V)
begin
  return the Euclidean distance from U to V;
end;
```

procedure project (R, U, V)

begin

return the projection of R onto the line defined by points U and V;

end;

It is possible for Z to equal X_k so that $Z - X_k = 0$. When this occurs, we perturb Z slightly in the y-coordinate direction. Also, since the threats overlap, we sometimes obtain a path with zig-zagging which intersects with one or more circles. At the conclusion of project, we call a scrubber procedure which removes points which appear inside a circle. This is illustrated in Figures 4 and 5.

Figures 4 and 5 about here

To obtain a path, we initialize $I_1, ..., I_{L+1}$ with a set of values, run projector followed by the scrubber. We compare the path obtained with the shortest feasible path found so far and retain the shortest feasible path. The

middle or direct path is obtained by setting I_1 to M, I_2 to T, and L to 1. The high path attempts to find a path such that all threats are below this path. For this path, I_2 is set to the center of the circle first encountered when we sweep clockwise a line segment anchored at M from the vertical position. Let X_h denote this point and set I_2 to X_h. This can be accomplished by examining the slopes of the line segments $[M, X_k]$ and selecting the one with the largest slope. The point I_3 is set to the X_K first encountered when we sweep counter-clockwise a line segment anchored at T from the vertical position. Finally, I_4 is set to T and L is set to 3. These starting segments generally result in a path with every threat below the path. A low path can be constructed in a similar manner by modifying the selection routine for I_2 and I_3. A middle high path is generated by selecting I_2 to be on the line segment $[M, X_h]$. A middle low path is constructed in a similar manner.

3.2 The Circumscribed Triangle Algorithm

The circumscribed triangle algorithm starts at M (the missile location) and constructs line segments by moving from a given segment point tangentially to the circumference of nearby threat regions, until a suitable straight-line

path from a segment point to T (the target location) is found. This basic strategy is embedded in a branch-and-bound framework, so that a minimal distance path of this restricted type is obtained.

In this algorithm we allow the missile to intersect a point on the circumference of a threat region but do not allow the missile to fly through any interior point of a threat region. Further, we allow a limit to be placed on the *turn angle* at any segment point, since a missile is unlikely to be making a sharp turn of, say, 90° or more at such a point. (Given segments $[Y_{i-1}, Y_i]$ and $[Y_i, Y_{i+1}]$, the turn angle at segment point Y_i is defined to be the complement of angle $Y_{i-1}Y_iY_{i+1}$.)

Let $d(E, F)$ denote the Euclidean distance between points E and F. The basic step in proceeding from segment point Y_i tangentially to a (nearby) threat region $R_k = \{s : d(X_k, s) \leq r_k\}$ involves constructing the circumscribing triangle on R_k having one vertex at Y_i (see Figure 6).

Figure 6 about here

The tangential points in Figure 6 are T_1 , T_2, and T_3 and the vertex points

are Y_i, V_1, and V_2. Sides Y_iV_1 and Y_iV_2 are the equal sides of the (isosceles) circumscribing triangle. Tangential point T_3 is easily computed since it lies on the line through Y_i and X_k with $d(X_k, T_3) = r_k$. To compute the other points we make use of a result from elementary geometry which states that the radius r of a circle inscribed in a triangle with sides a, b, and c is given by

$$r = \frac{\sqrt{s(s-a)(s-b)(s-c)}}{s}$$

where $s = \frac{1}{2}(a+b+c)$.

Letting $a = d(Y_i, V_1)$, $b = d(Y_i, V_2)$, and $c = d(V_1, V_2)$ with $a = b$, a one-dimensional search on the value of c can be used to produce a value of $r = r_k$. From this c value, points V_1 and V_2 can be determined. Points T_1 and T_2 can then be computed as the nearest points to X_k on Y_iV_1 and Y_iV_2, respectively.

We intend to make use of these tangential lines to construct the next segment, if possible. Before using points on either tangential line, we first make sure that the respective turn angles at Y_i would be within the turn angle limit.

The points V_1 and V_2 have a special property. Each is the first point on the

tangential line from Y_i through itself which has a complete 180° field of view (ignoring turn angle constraints) of the side of R_k opposite Y_i. By contrast, tangent points T_1 and T_2 each have less than a 90° field of view (ignoring turn angle constraints) of that side (see Figure 7). With this in mind V_1 (or V_2) would seem to be a good choice for the end point of a next segment in our path construction. We actually attempt to use a point between T_1 and V_1 (or between T_2 and V_2) for the next end point. In general, such a point can be represented as a convex combination $Y_{i+1} = (1-\alpha)T_1 + \alpha V_1$ (or $Y_{i+1} = (1-\alpha)T_2 + \alpha V_2$), for some $0 \leq \alpha \leq 1$. In order for such a point Y_{i+1} to be usable, we must insure that the segment $[Y_i, Y_{i+1}]$ is *clear*. (A line segment is defined to be clear if it does not intersect the interior of any of the threat regions.)

We first search for a minimum value of α with $0 \leq \alpha \leq 1$ such that $[Y_i, (1-\alpha)T_1 + \alpha V_1]$ (or $[Y_i, (1-\alpha)T_2 + \alpha V_2]$) is clear and $[(1-\alpha)T_1 + \alpha V_1, T]$ (or $[(1-\alpha)T_2 + \alpha V_2, T]$) is clear without exceeding the turn angle limit at $(1-\alpha)T_1 + \alpha V_1$ (or $(1-\alpha)T_2 + \alpha V_2$). If such a point is found we have a *feasible completion* by adding the two segments $[Y_i, Y_{i+1} = (1-\alpha)T_1 + \alpha V_1]$ and $[Y_{i+1} = (1-\alpha)T_1 + \alpha V_1, Y_{i+2} = T]$ (or $[Y_i, Y_{i+1} = (1-\alpha)T_2 + \alpha V_2]$ and $[Y_{i+1} = (1-\alpha)T_2 + \alpha V_2, Y_{i+2} = T]$).

If such a point leading to a feasible completion cannot be found, it would seem natural to use V_1 (or V_2) by default, due to its wide field of view. However, in practice this leads to longer paths and we find it better to use a point between T_1 and V_1 (or between T_2 and V_2) for the next end point. We have parameterized this by defining a *backoff percentage* β which specifies that, in this case, Y_i is to be chosen as the convex combination $(1-\beta)T_1+\beta V_1$ (or $(1-\beta)T_2+\beta V_2$). Experimentally, we have found $\beta = \frac{1}{2}$ to be a good choice.

Figure 7 about here

The branch-and-bound algorithm we have developed based on this circumscribed triangle approach is given as follows:

Algorithm : *circumscribe*

Inputs :

M	- missile location
T	- target location
K	- number of threats

$X_1, ..., X_K$ - center points of the threats

$r_1, ..., r_K$ - radii of the threats

Z - turn angle limit (degrees)

β - fall back parameter ($0 < \beta \leq 1$)

Outputs :

D - length of best constructed solution path

S - number of segments in best constructed solution path

$L_1, ..., L_S$ - segments of best constructed solution path

Constructs :

B - pointer to last point in best constructed solution path

Q - branch-and-bound queue

$d(E, F)$ - distance from E to F

$L(E, F)$ - line segment from E to F

$R(C, r)$ - threat region with center C and radius r

$S(C, r)$ - interior of threat region with center C and radius r

I - number of points constructed

Y_i - i^{th} constructed point

B_i - pointer to point previous to Y_i

D_i - path distance prior to Y_i

S_i - path segments prior to Y_i

E_i - set of threat points eligible for moving away from point Y_i

T_1, T_2, T_3 - constructed tangent points of circumscribled triangle

V_1, V_2 - constructed vertices of circumscribled triangle

begin

$D \leftarrow \infty$; $S \leftarrow 0$; $B \leftarrow 0$;

if $L(M,T) \cap S(X_i, r_i) = \Phi$ for $i = 1..K$ **then**

$D \leftarrow d(M,T)$; $S \leftarrow 1$; $L_1 \leftarrow [M,T]$;

else

begin

$I \leftarrow 1$; $Y_1 \leftarrow M$; $B_1 \leftarrow 0$; $D_1 \leftarrow 0$; $S_1 \leftarrow 0$; $E_1 \leftarrow \{R_1, ..., R_K\}$;

put $(Y_1, B_1, D_1, S_1, E_1)$ on Q;

while $Q \neq \Phi$ **do**

begin

remove some $(Y_i, B_i, D_i, S_i, E_i)$ from Q;

$b \leftarrow B_i$;

select some R_j from E_i; $E_i \leftarrow E_i \setminus R_j$;

if $E_i \neq \Phi$ **then** put $(Y_i, B_i, D_i, S_i, E_i)$ back on Q;

form the triangle $Y_iV_1V_2$ circumscribing R_j having

its tangent points $T_1 \in Y_iV_1, T_2 \in Y_iV_2, T_3 \in V_1V_2$;

for $k = 1$ **to** 2 **do**

begin

if $180 - Y_bY_iT_k \leq Z$ and

$L(Y_i, T_k) \cap S(X_l, r_l) = \Phi$ for $l = 1..K$ and

$D_i + d(Y_i, T_k) + d(T_k, T) < D$ **then**

begin

search for a point U on $L(T_k, V_k)$ nearest to T_k satisfying:

$180 - Y_iUT \leq Z$ and

$L(Y_i, U) \cap S(X_l, r_l) = \Phi$ for $l = 1..K$ and

$L(U, T) \cap S(X_l, r_l) = \Phi$ for $l = 1..K$;

if such a point U exists **then**

begin

if $D_i + d(Y_i, U) + d(U, T) < D$ **then**

begin

$I \leftarrow I + 1;$

$Y_I \leftarrow U;\ B_I \leftarrow i;\ D_I \leftarrow D_i + d(Y_i, U);\ S_I \leftarrow S_i + 1;$

$m \leftarrow I;\ I \leftarrow I + 1;$

$Y_I \leftarrow T;\ B_I \leftarrow m;\ D_I \leftarrow D_m + d(U, T);\ S_I \leftarrow S_i + 2;$

$D \leftarrow D_I;\ S \leftarrow S_I;\ B \leftarrow I;$

end;

else

begin

if $E_i \neq \Phi$ **then**

begin

$U \leftarrow (1 - \beta)T_k + \beta V_k;$

if $L(Y_i, U) \cap S(X_i, r_l) = \Phi$ for $l = 1..K$ **then**

begin

$I \leftarrow I + 1;$

$Y_I \leftarrow U;\ B_I \leftarrow i;\ D_I \leftarrow D_i + d(Y_i, U);\ S_I \leftarrow S_i + 1;\ E_I \leftarrow E_i;$

put $(Y_I, B_I, D_I, S_I, E_I)$ on Q;

end;

end;

end;

end;

end;

end;

$i \leftarrow S$; $k \leftarrow B$;

while $k \neq 0$ do $j \leftarrow B_k$; $L_i \leftarrow [Y_j, Y_k]$; $i \leftarrow i - 1$; $k \leftarrow j$;

end;

Figure 8 shows an optimal path through a region containing three threats generated by the circumscribed triangle algorithm. Figure 9 shows additional partial paths produced by the branch-and-bound process contained in the algorithm. The actual implementation, whose results are reported in Section IV, actually makes two applications of the algorithm, the second of which has the roles of missile and target reversed. Of course, any path obtained with

missile and target reversed would itself need to be reversed. The shortest of the two paths generated is selected as the best path.

Figures 8 and 9 about here

---

4 EMPIRICAL ANALYSIS

Using a grid size of 780 x 700, we randomly generated five groups of test problems. Each group had five SAM sites with radii randomly generated on the interval [50, 150]. The four problems in each group had identical SAM sites, but different missile and target locations with the missiles located on the left boundary and targets located on the right boundary. The missile and target locations were randomly selected and were different for each of the twenty test problems. The input for each problem consists of seven points (the missile location, the target location, five SAM sites) and five radii.

A graphical system was developed using Tcl and Tk (see Welch [6]). Tcl stands for Tool Command Language and Tk is a toolkit for window programming. The system allowed us to display the problem graphically and

construct a path manually by simply pointing and clicking. This system was used to manually construct solutions for each of the twenty test problems. Both the projection and the circumscribed triangle algorithms were also applied to each of the test problems. The results are summarized in Table 1. The manual paths tended to steer clear of the SAM sites and use only a few segments at the penalty of substantially longer paths. The projection algorithm generated relatively short paths at the expense of substantially more segments. The circumscribed triangle algorithm worked best, using a small number of segments and producing very short paths. The paths obtained manually and by the projection algorithm may be found in Figures 10 through 14 in Appendix A. The paths obtained by the circumscribed triangle algorithm may be found in Figures 15 through 19 in Appendix B.

Table 1. Comparison of Three Procedures for the Mission Planning Problem

	Manual		Projection Alg		Circumscribed Triangle Alg	
	Distance	Segments	Distance	Segments	Distance	Segments
1	1000	2	959	3	957	3
2	1002	3	853	5	853	3
3	998	4	948	5	853	4
4	1040	3	946	4	945	3
5	826	2	798	2	795	2
6	934	4	881	4	878	4
7	1097	2	1043	4	1042	3
8	813	2	807	2	805	2
9	802	2	784	2	782	2
10	1107	3	1054	4	1049	3
11	870	3	852	3	847	3
12	839	2	823	2	820	2
13	822	2	811	2	810	2
14	1030	3	946	5	945	3
15	1039	3	961	5	966	3
16	877	3	839	3	837	3
17	911	3	864	4	865	3
18	1064	3	1014	4	1009	3
19	1013	4	974	4	900	4
20	900	3	855	3	862	2
Totals	18,984	56	18,012	70	17,820	57
Scaled	1.07	0.98	1.01	1.23	1.0	1.0

5 SUMMARY AND CONCLUSIONS

This manuscript presents two algorithms for the mathematical problem of finding a path consisting of line segments from a given origin to a given destination which does not pass through any of a set of threats represented by circles. Neither algorithm requires the use of a grid imposed on the military theatre which is required by the competing algorithms of which we are aware.

The projection algorithm initially ignores the threats and constructs a path from the origin to the destination. Let $[Y_i, Y_{i+1}]$ denote the line segment defined by endpoints Y_i and Y_{i+1}. If $[Y_i, Y_{i+1}]$ passes through a threat whose center is X, then X is projected onto $[Y_i, Y_{i+1}]$ to obtain Z. It next finds a point W outside the threat on the line defined by Z and X. The original segment $[Y_i, Y_{i+1}]$ is replaced by two segments $[Y_i, W]$ and $[W, Y_{i+1}]$. Continuing in this way usually leads to a feasible path.

The circumscribed triangle algorithm constructs triangles around the threats and selects break points on either the upper or lower edge of these circumscribed triangles. Hence, from the origin there are two potential paths around a threat (the upper path and the lower path). From each of these paths there may be additional threats from which we consider both the up-

per path and the lower path. This combinatorial feature of the strategy is handled in a branch-and-bound framework.

In an empirical analysis with twenty randomly generated test problems, we found that the circumscribed triangle algorithm consistently produced paths which were superior to those obtained either manually or by the projection algorithm. Also, our software implementation is so fast that this algorithm can be run on a PC in a real-time environment.

References

[1] A. Boroujerdi, C. Dong, Q. Ma, and B. Moret, "Joint Routing in Networks," undated technical report, Department of Computer Science, University of New Mexico, Albuquerque, NM.

[2] E. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik* 1 (1959) 269 - 271.

[3] R. Helgason, J. Kennington, and K. Lewis, "Finding Safe Paths in Networks," Technical Report 96-CSE-7, Department of Computer Science and Engineering, SMU, Dallas, TX 75275-0122 (1996).

[4] J. Solka, J. Perry, B. Poellinger, and G. Rogers, "Fast Computation of Optimal Paths Using a Parallel Dijkstra Algorithm with Embedded Constraints," *Neurocomputing* 8, (1995) 195 - 212.

[5] J. Solka, J. Perry, B. Poellinger, and G. Rogers, "Autorouting Using a Parallel Dijkstra Algorithm with Embedded Constraints," undated technical report, The Naval Surface Warfare Center, Dahlgren, VA.

[6] B. Welch, *Practical Programming in Tcl and Tk*, Prentice-Hall, Inc., Upper Saddle River, NJ (1995).

[7] M. Zuniga and P. Gorman, "Threat Site Overflight Modeling for Strike Route Optimization," undated technical report, Naval Research Laboratory, Washington, D.C.

[8] M. Zuniga, J. Uhlmann, and J. Hofmann, "The Interdependent Joint Routing Problem: Description and Algorithmic Approach," undated technical report, Naval Research Laboratory, Washington, D.C.

APPENDIX A

SOLUTIONS USING PROJECTION ALGORITHM

This appendix presents a graphical display of the twenty randomly generated test problems along with the mission obtained manually and the mission obtained by the projection algorithm. The black line is the mission obtained using *projector* and the gray line is the mission obtained manually.

Figures 10 to 14 about here

APPENDIX B

SOLUTIONS USING THE CIRCUMSCRIBED TRIANGLE ALGORITHM

This appendix presents a graphical display of the solutions produced by the circumscribed triangle algorithm.

Figures 15 to 19 about here

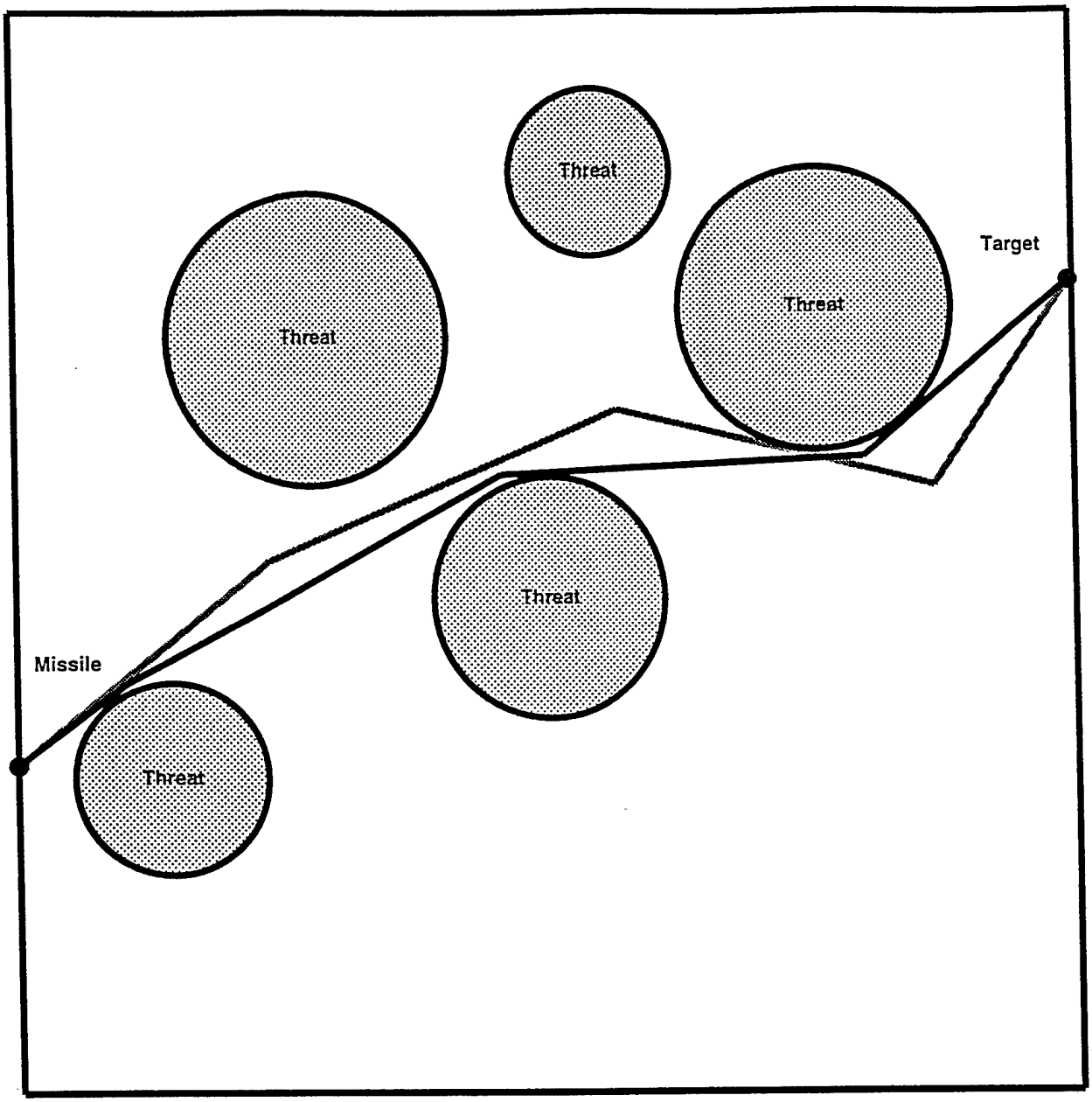

Figure 1: Illustration of a Feasible Mission Plan Through a Theatre Having Five SAM Sites

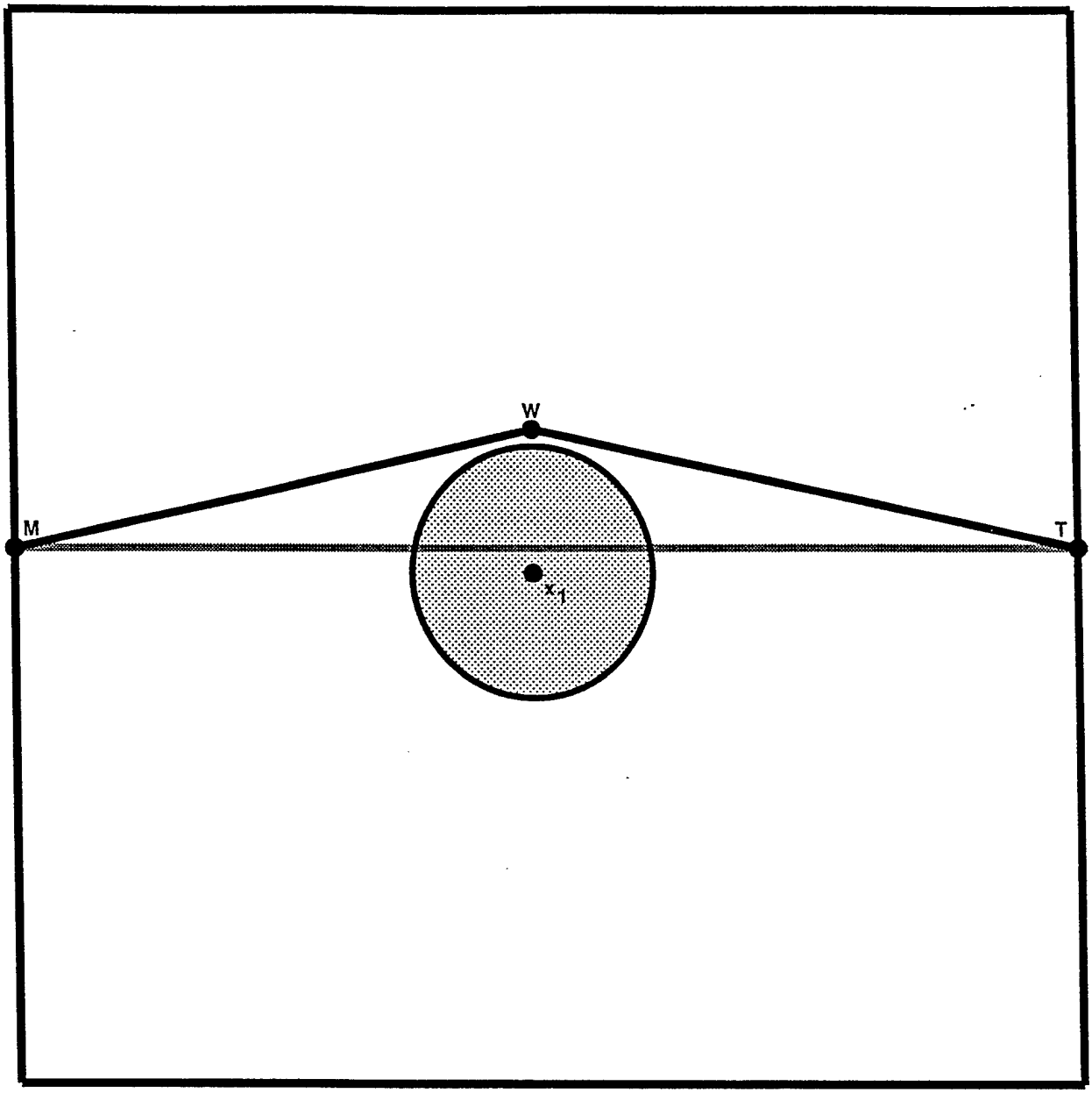

Figure 2: Deflection of a Path Around a Threat

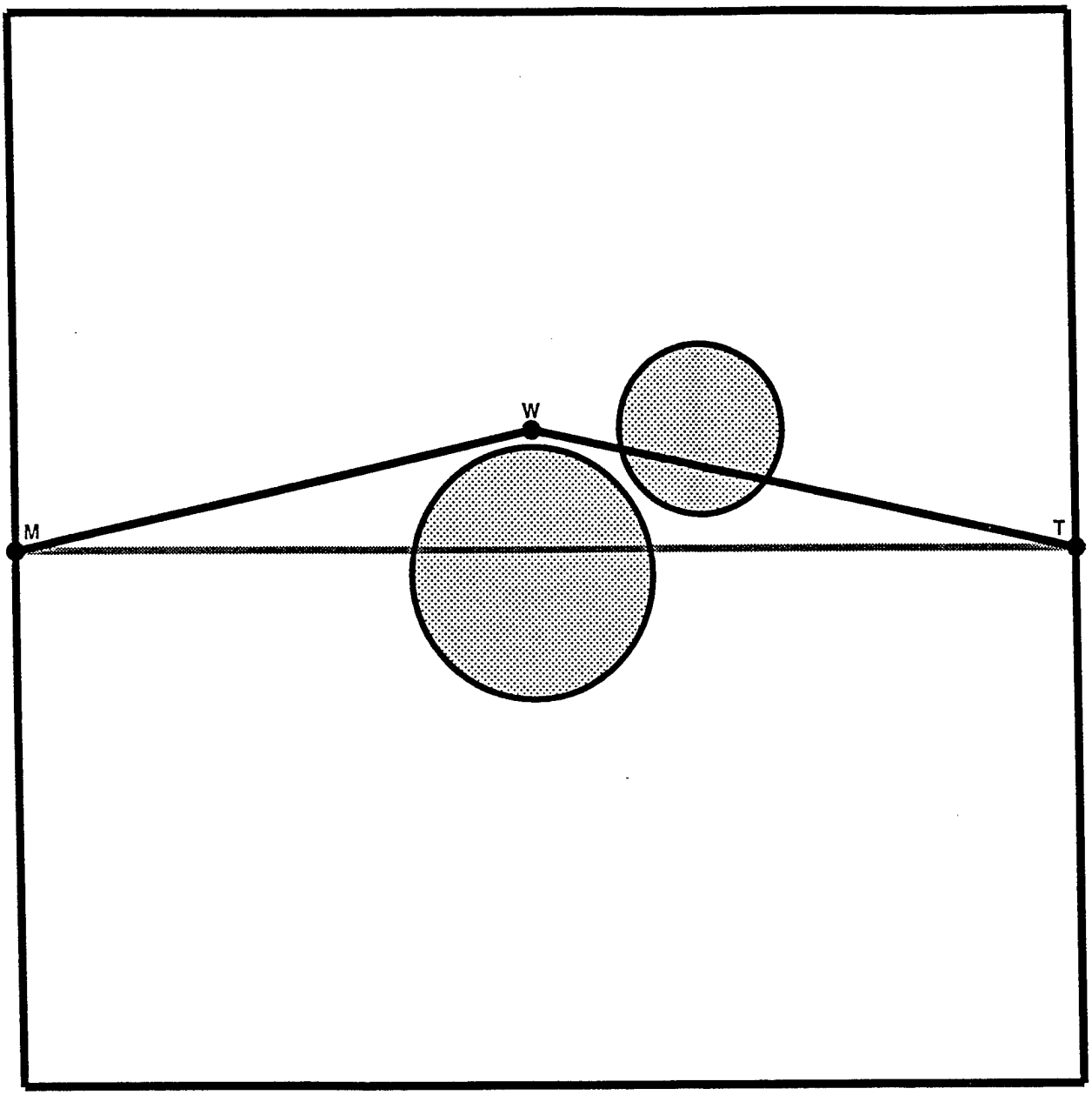

Figure 3: Path Deflection Which Intersects with a Different Threat

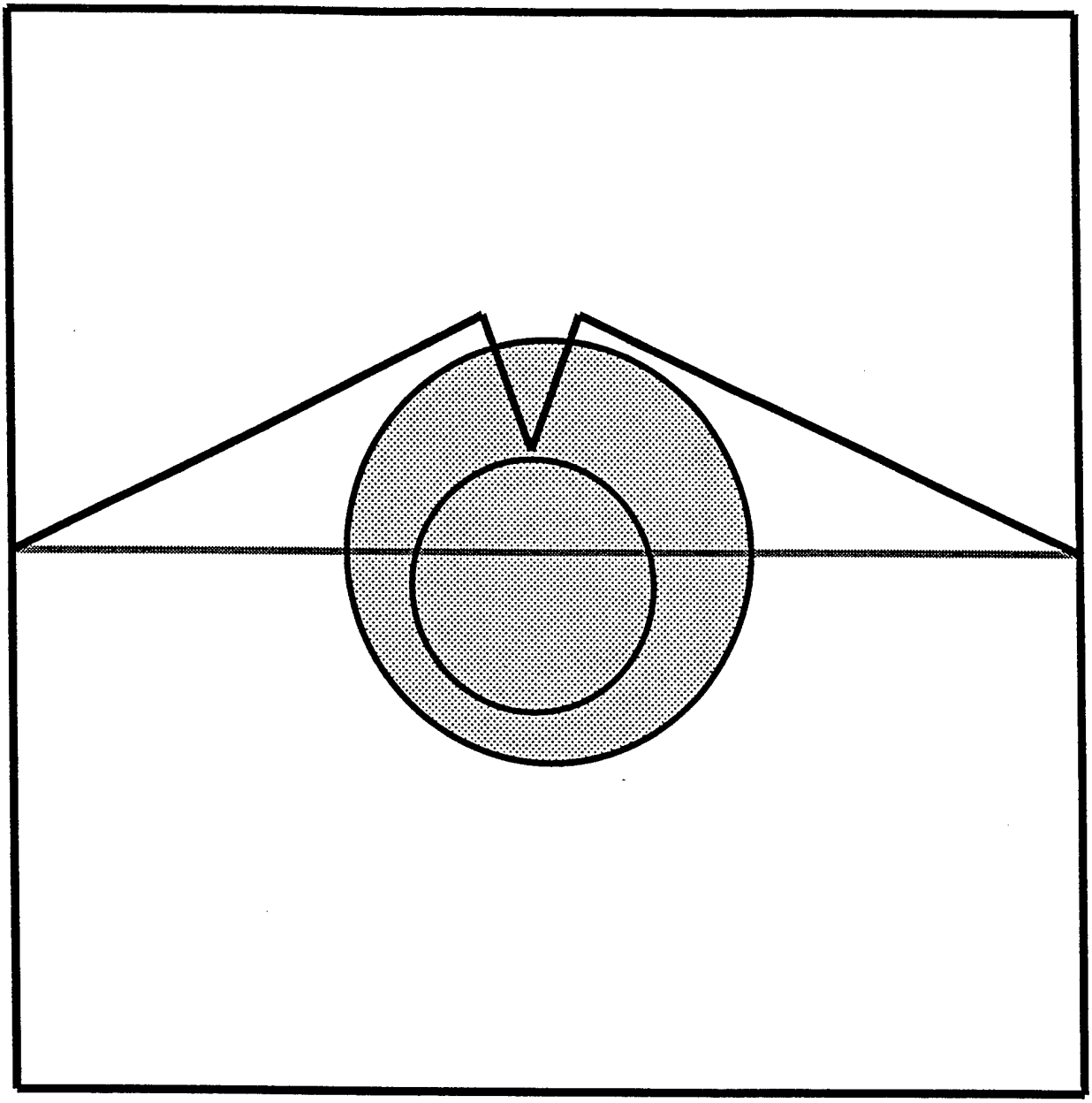

Figure 4: Results from Find Path for Overlapping Circles

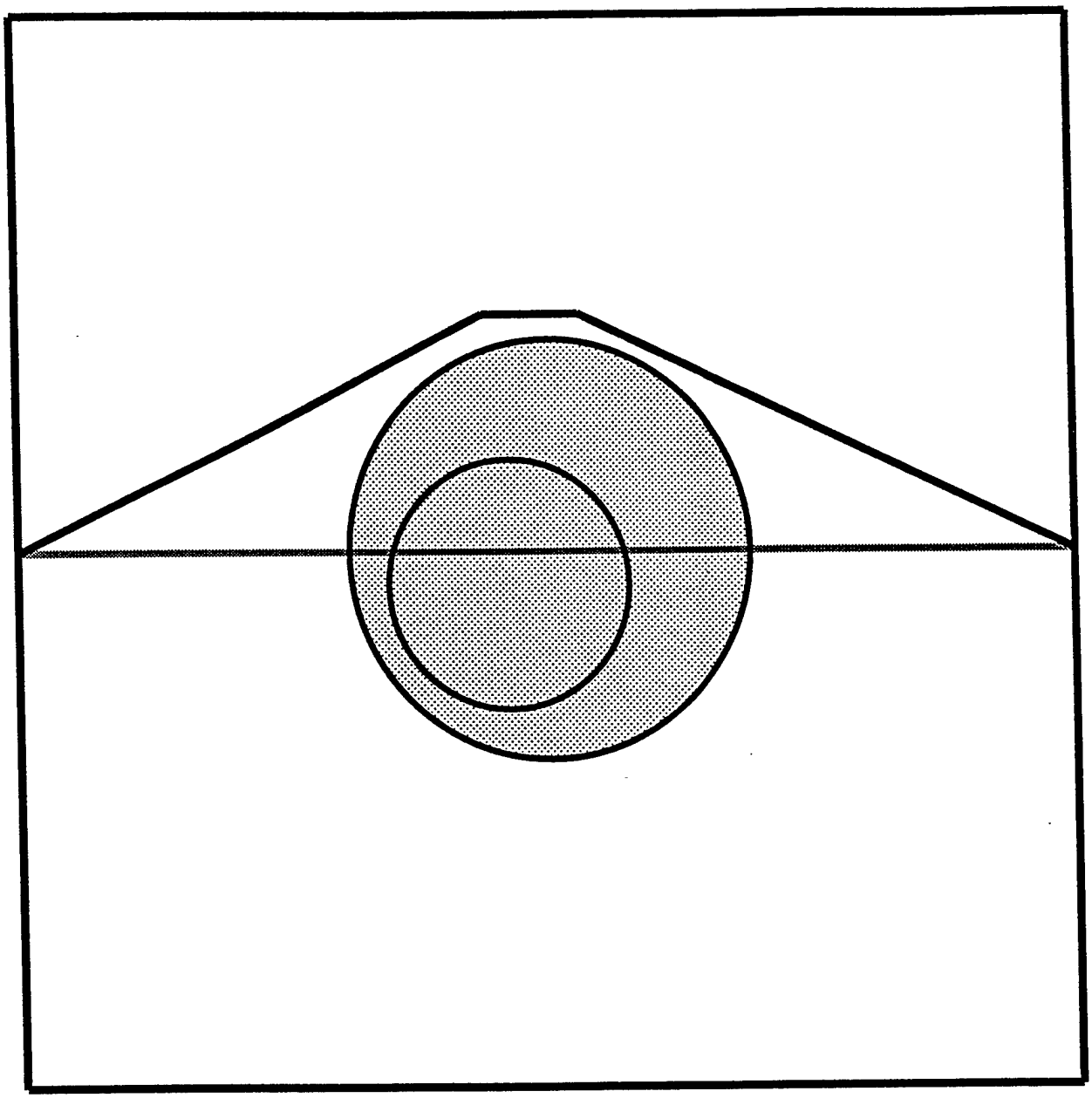

Figure 5: Path Obtained After Scrubbing the Path in Figure 4

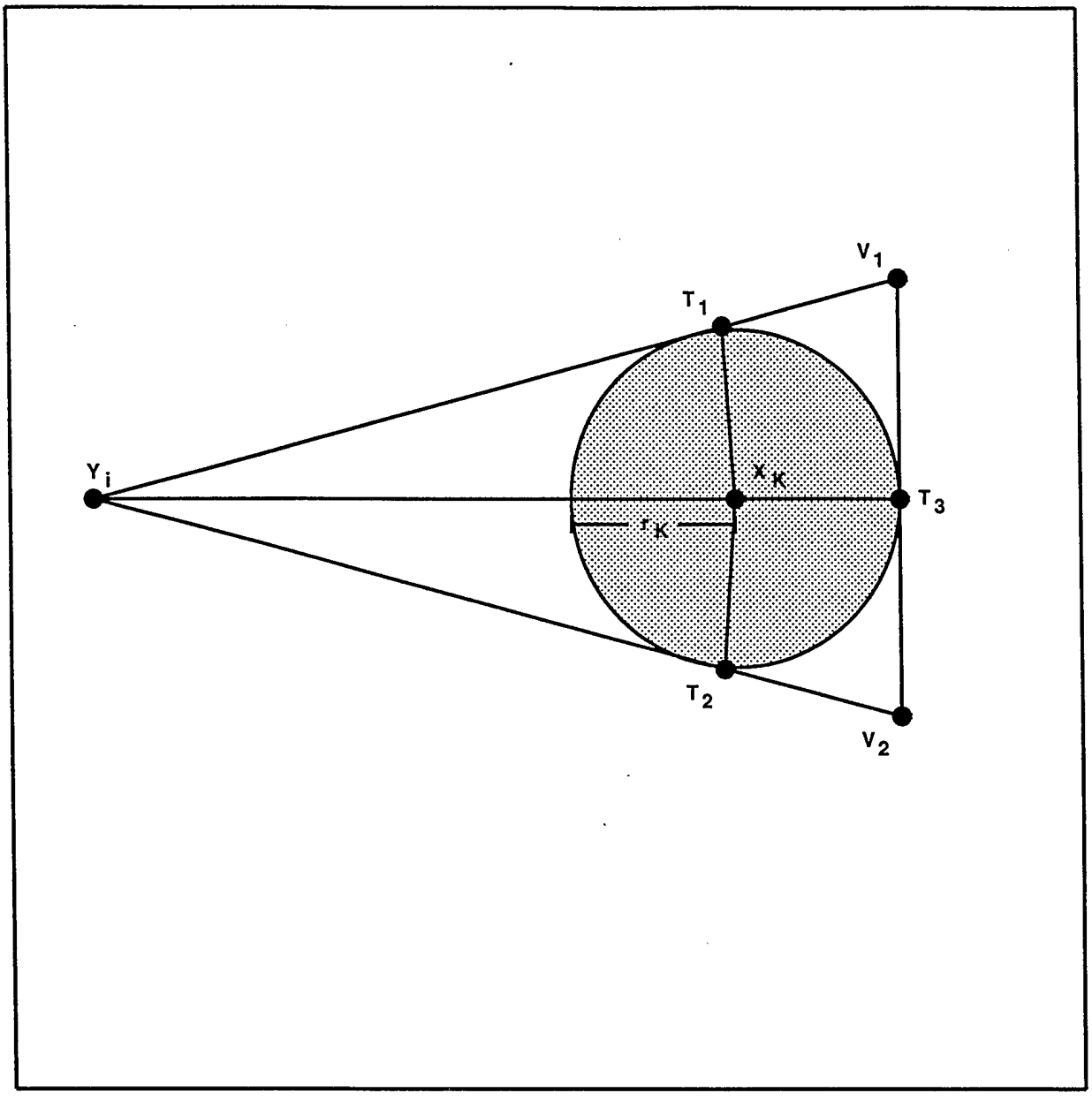

Figure 6: Circumscribing Triangle With Vertex Y_i For Threat Region R_k

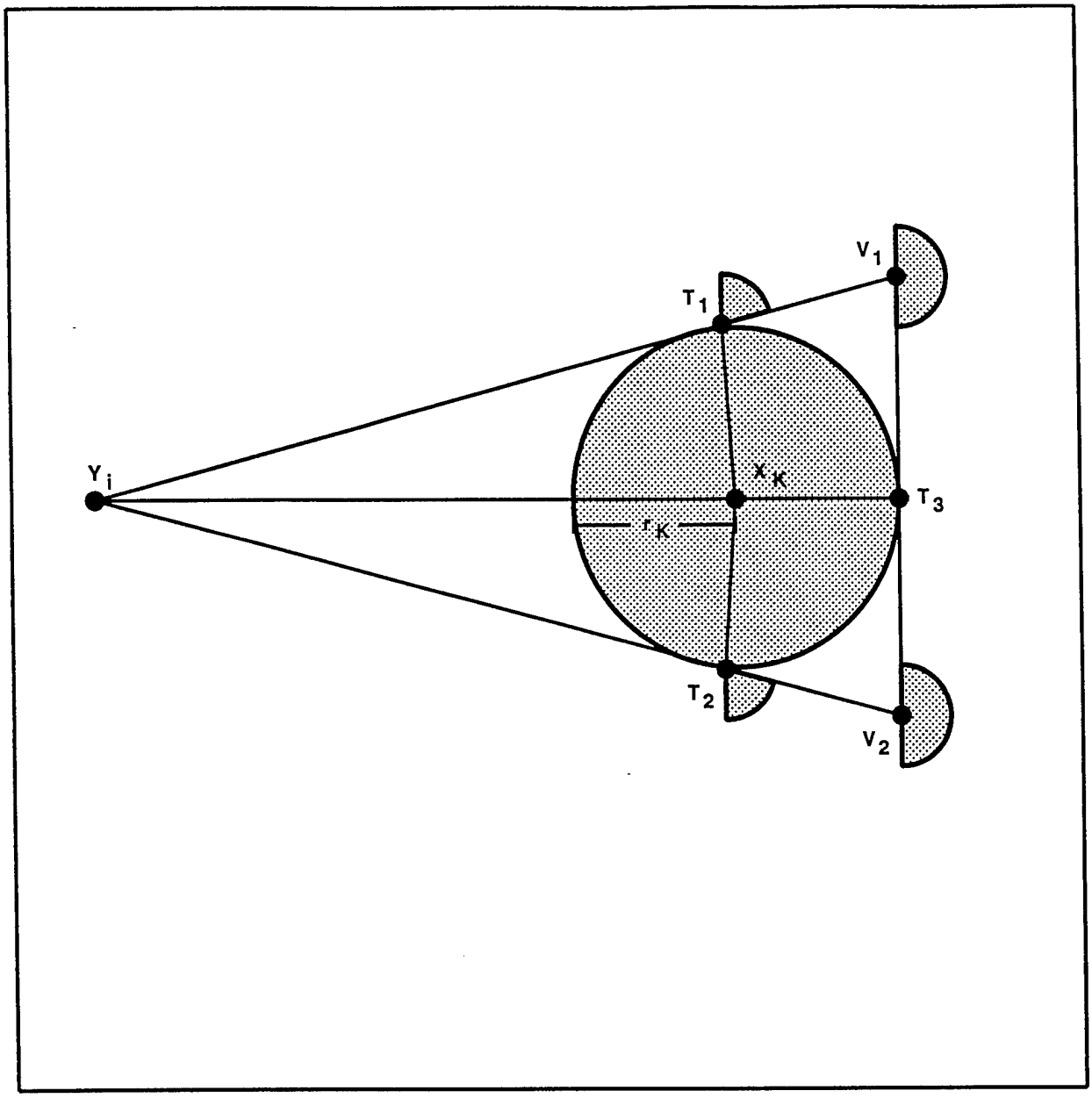

Figure 7: Opposite Side Field Of View At Points V_1, V_2, T_1, T_2

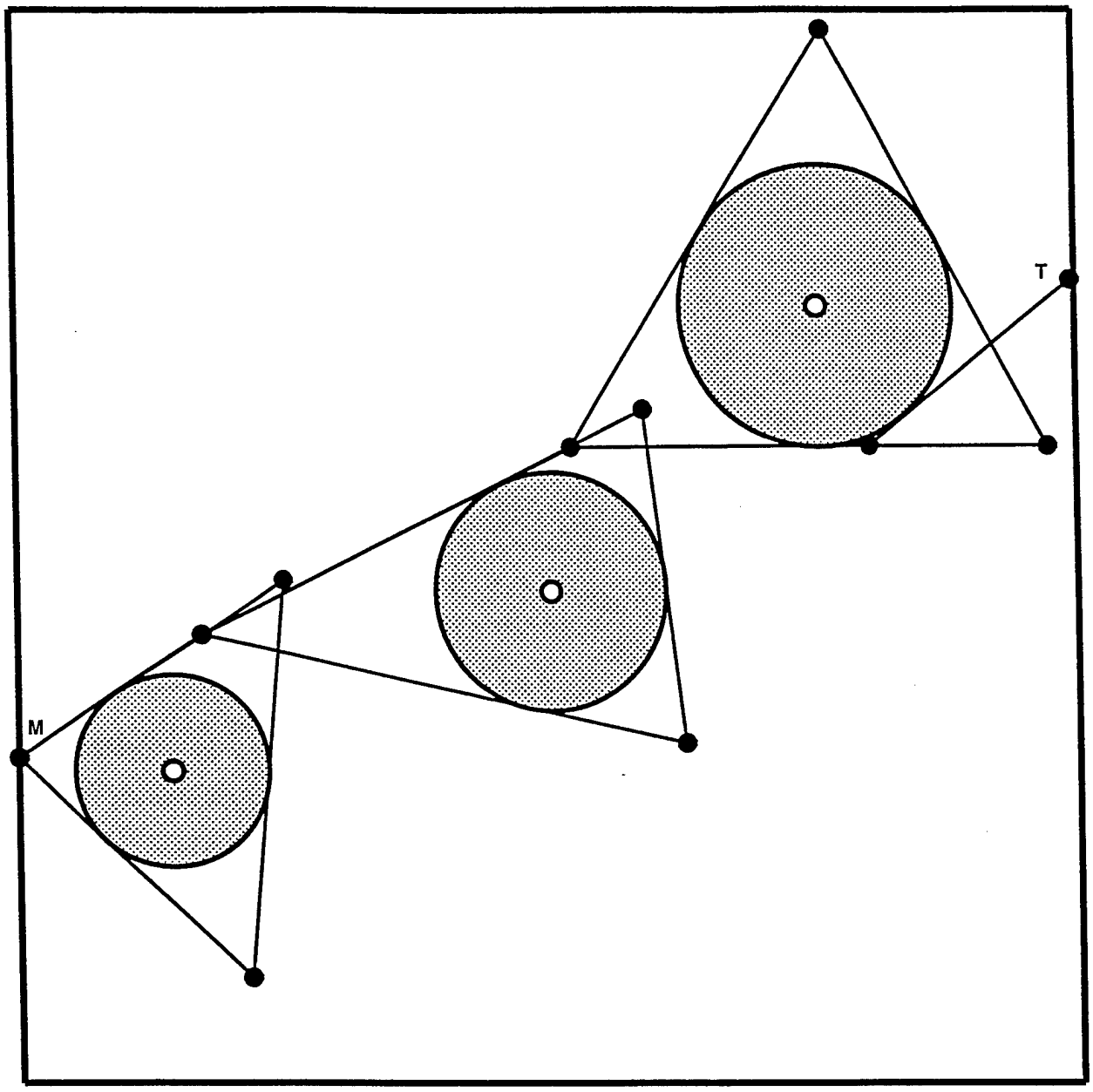

Figure 8: Path Produced by the Circumscribed Triangle Algorithm

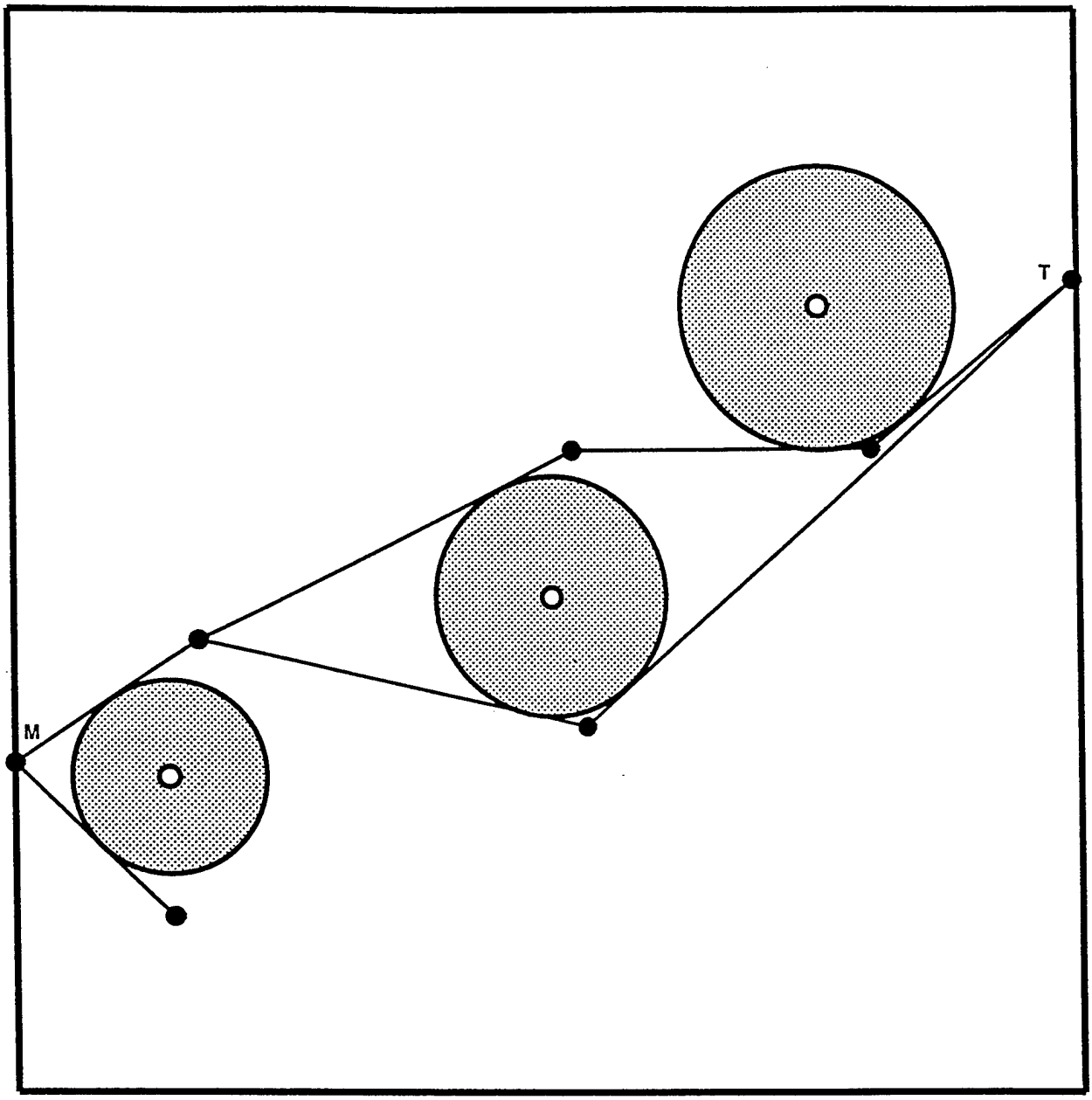

Figure 9: Partial Paths Considered by the Circumscribed Triangle Algorithm

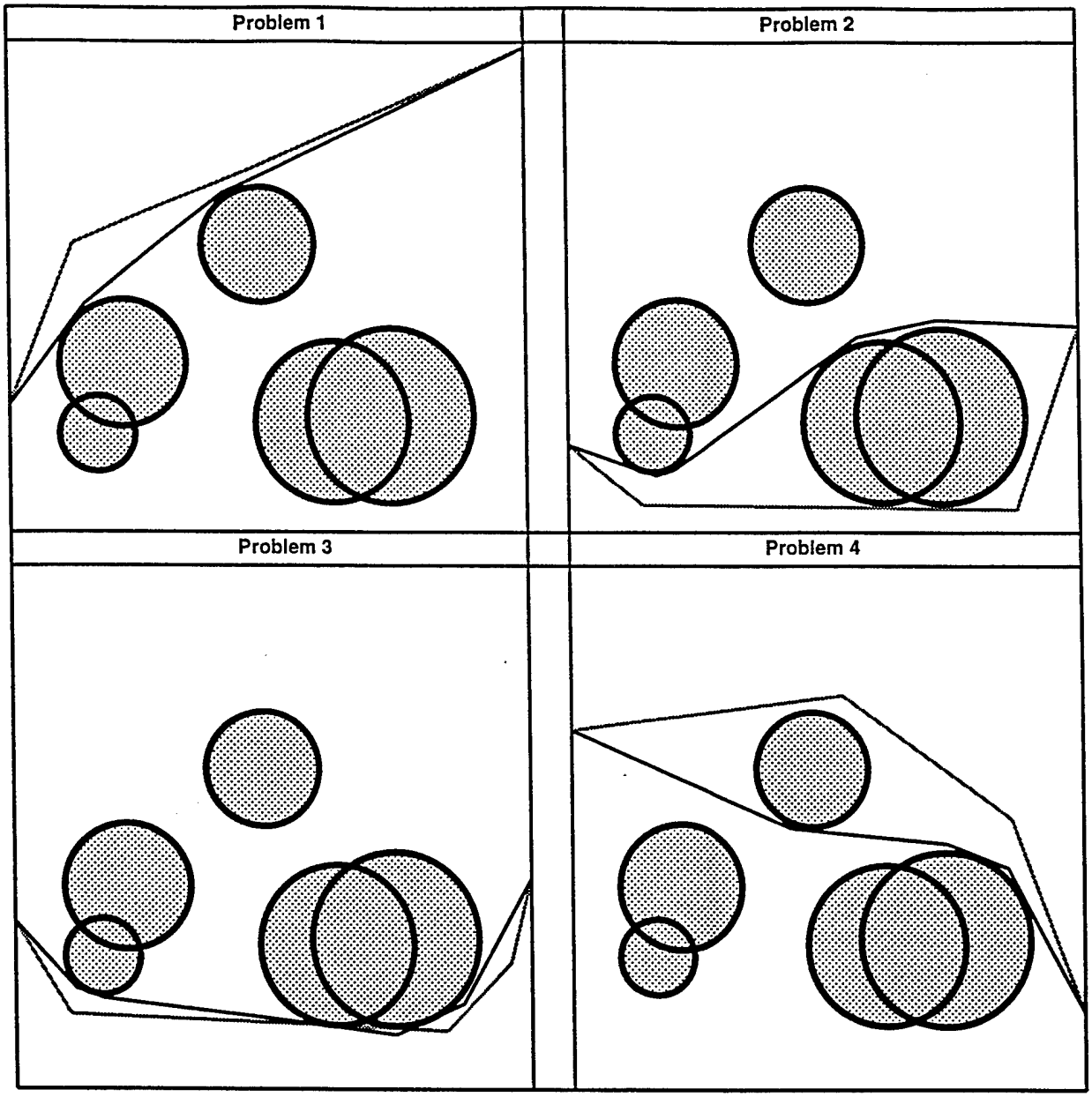

Figure 10: Display of Problems 1, 2, 3, and 4

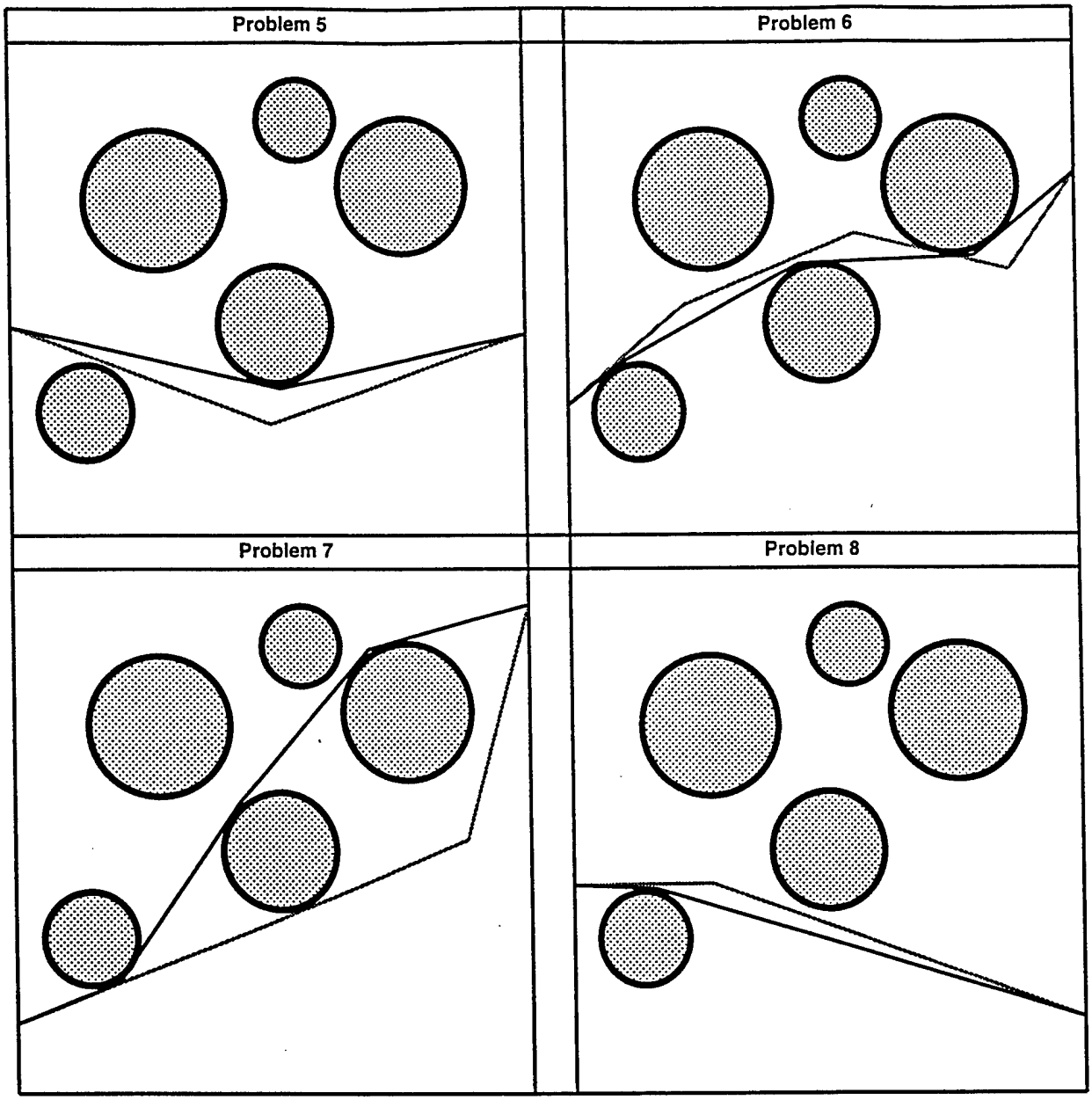

Figure 11: Display of Problems 5, 6, 7, and 8

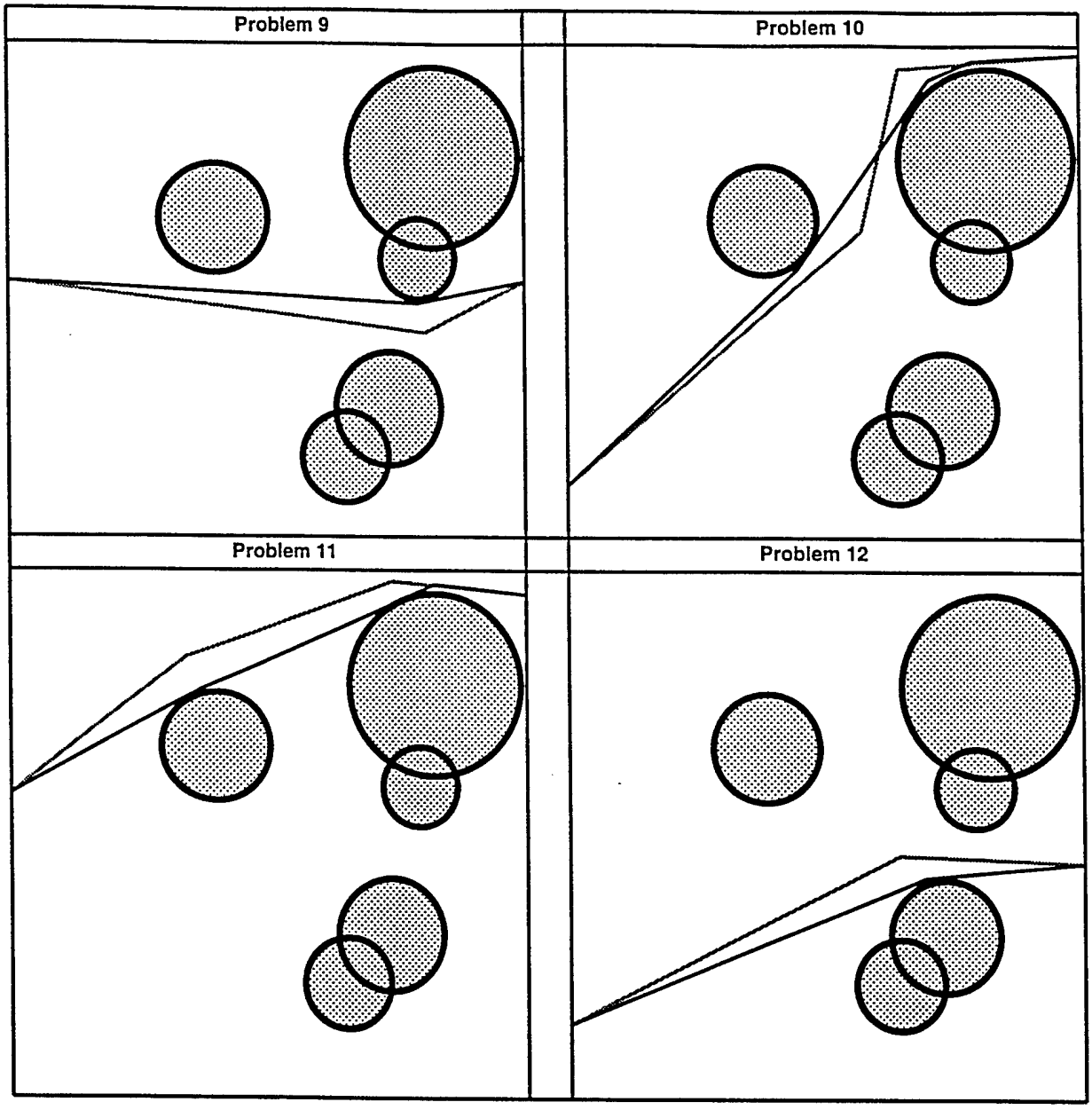

Figure 12: Display of Problems 9, 10, 11, and 12

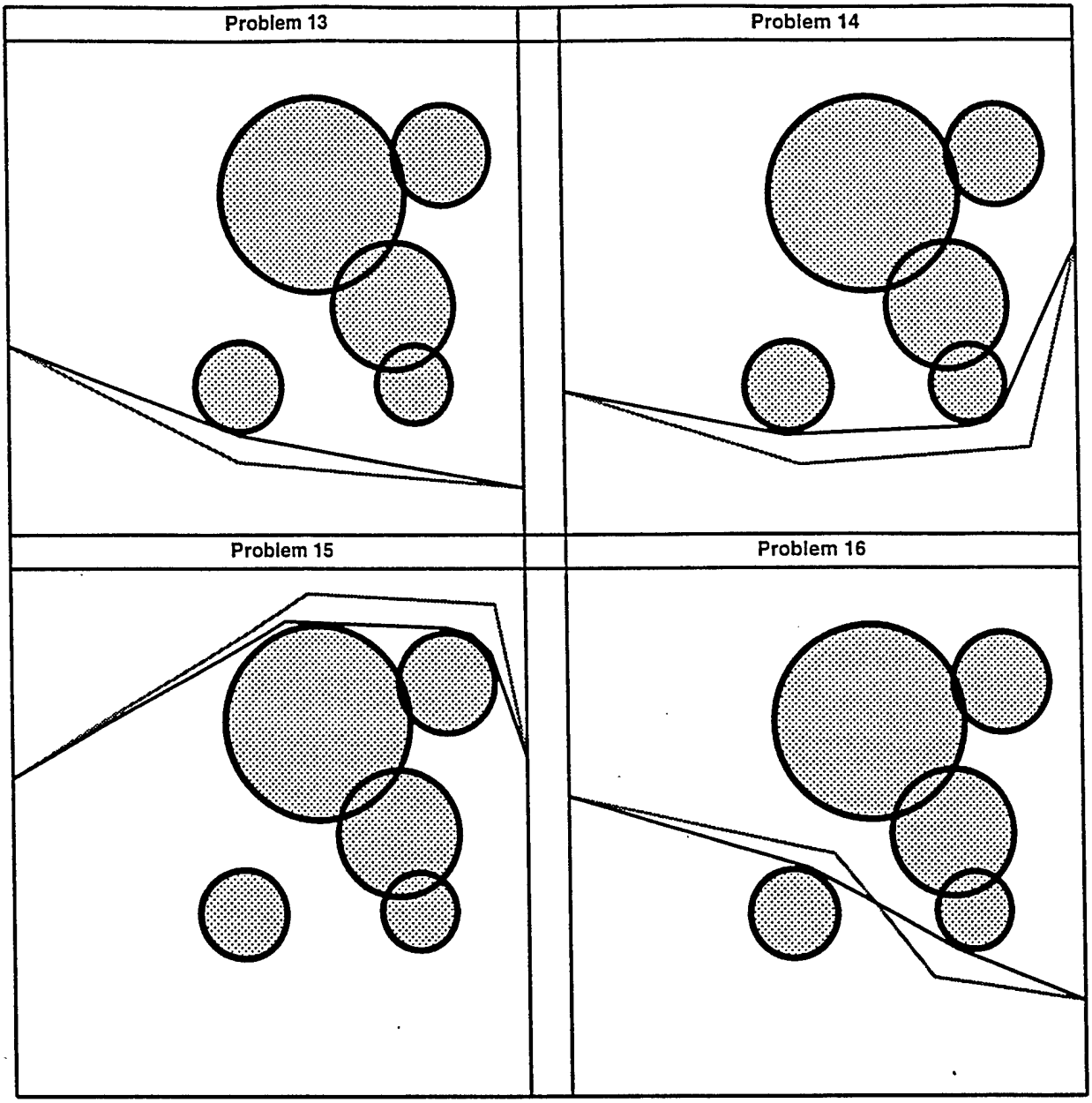

Figure 13: Display of Problems 13, 14, 15, and 16

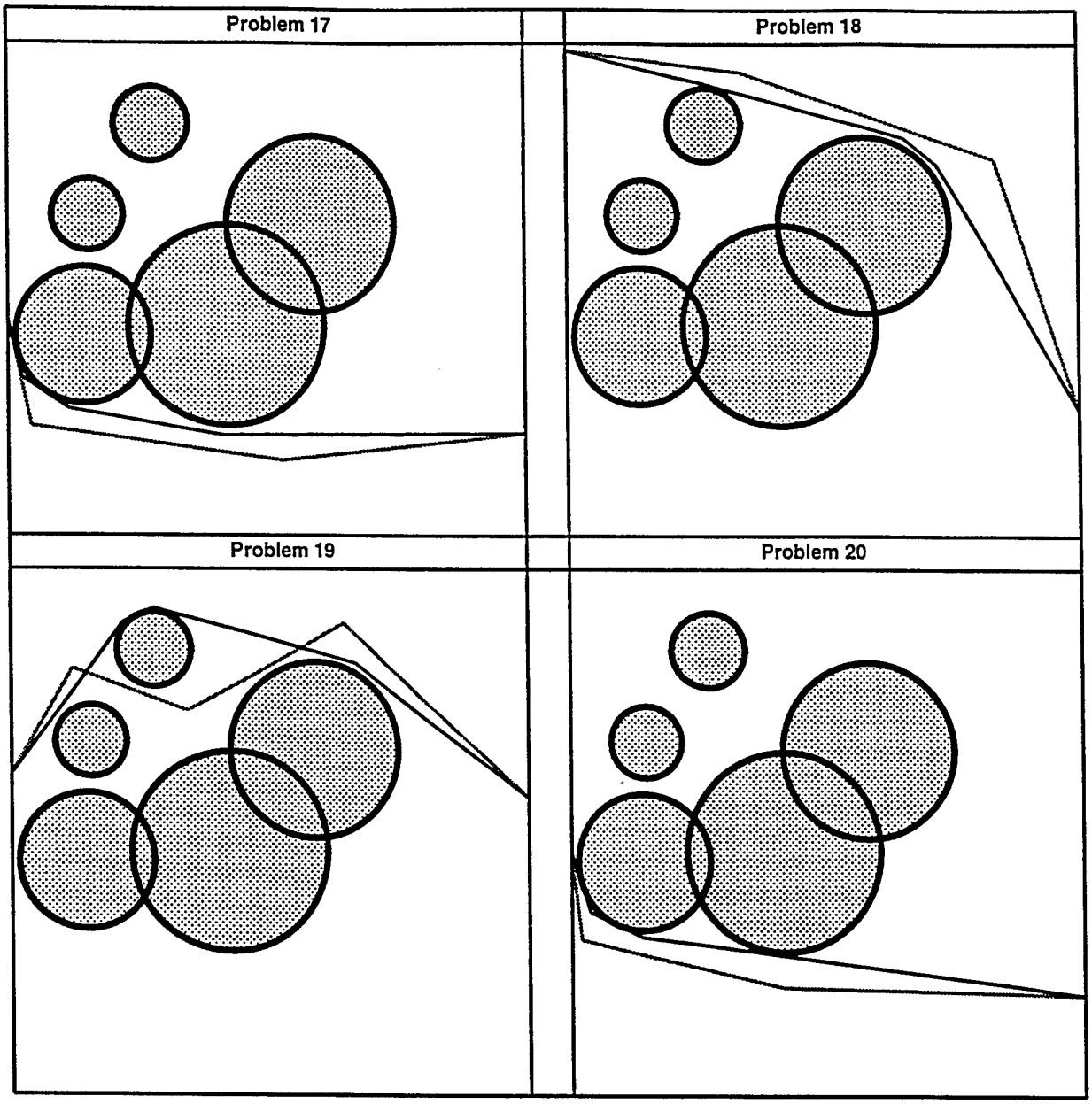

Figure 14: Display of Problems 17, 18, 19, and 20

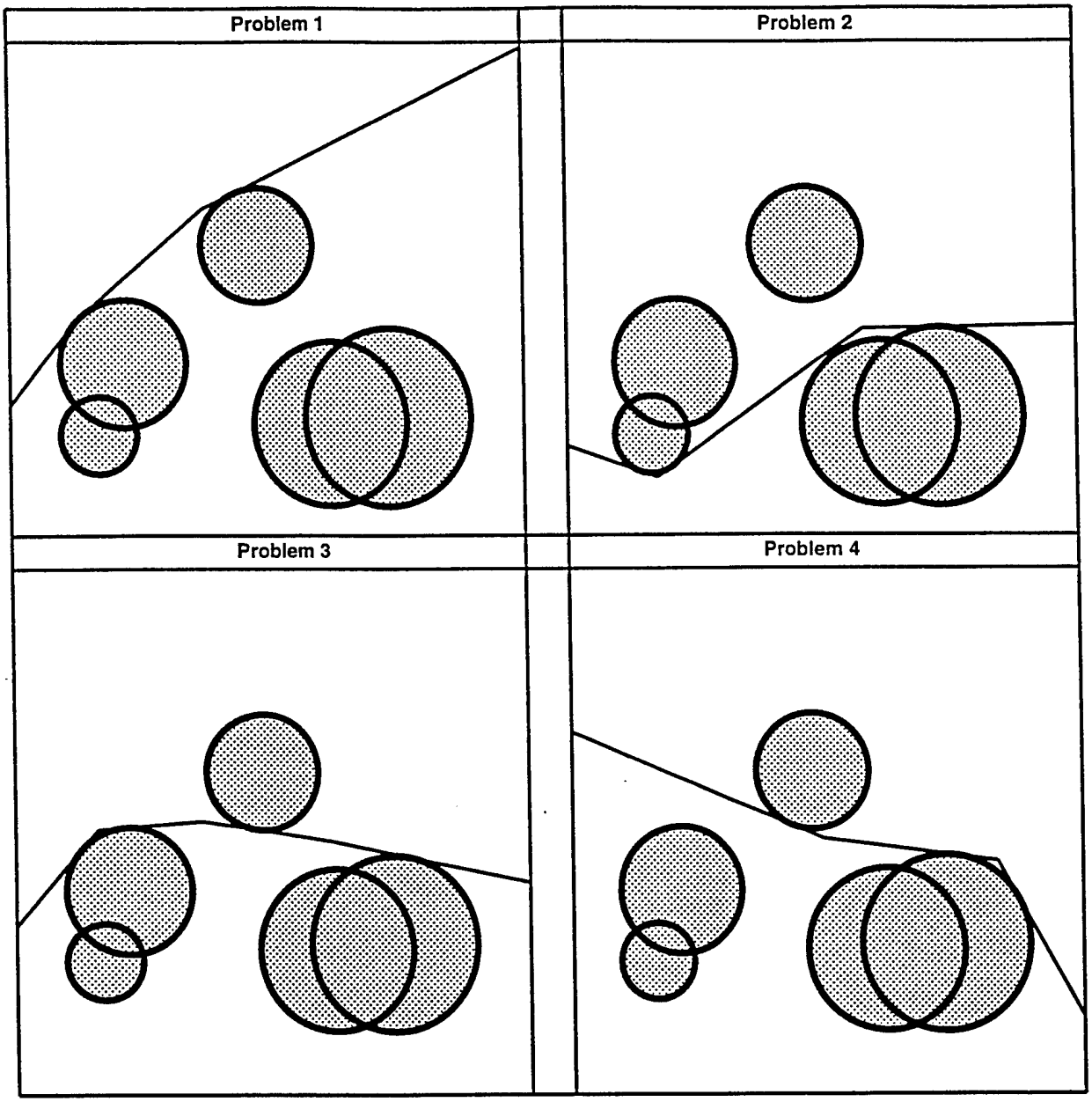

Figure 15: Display of Problems 1, 2, 3, and 4

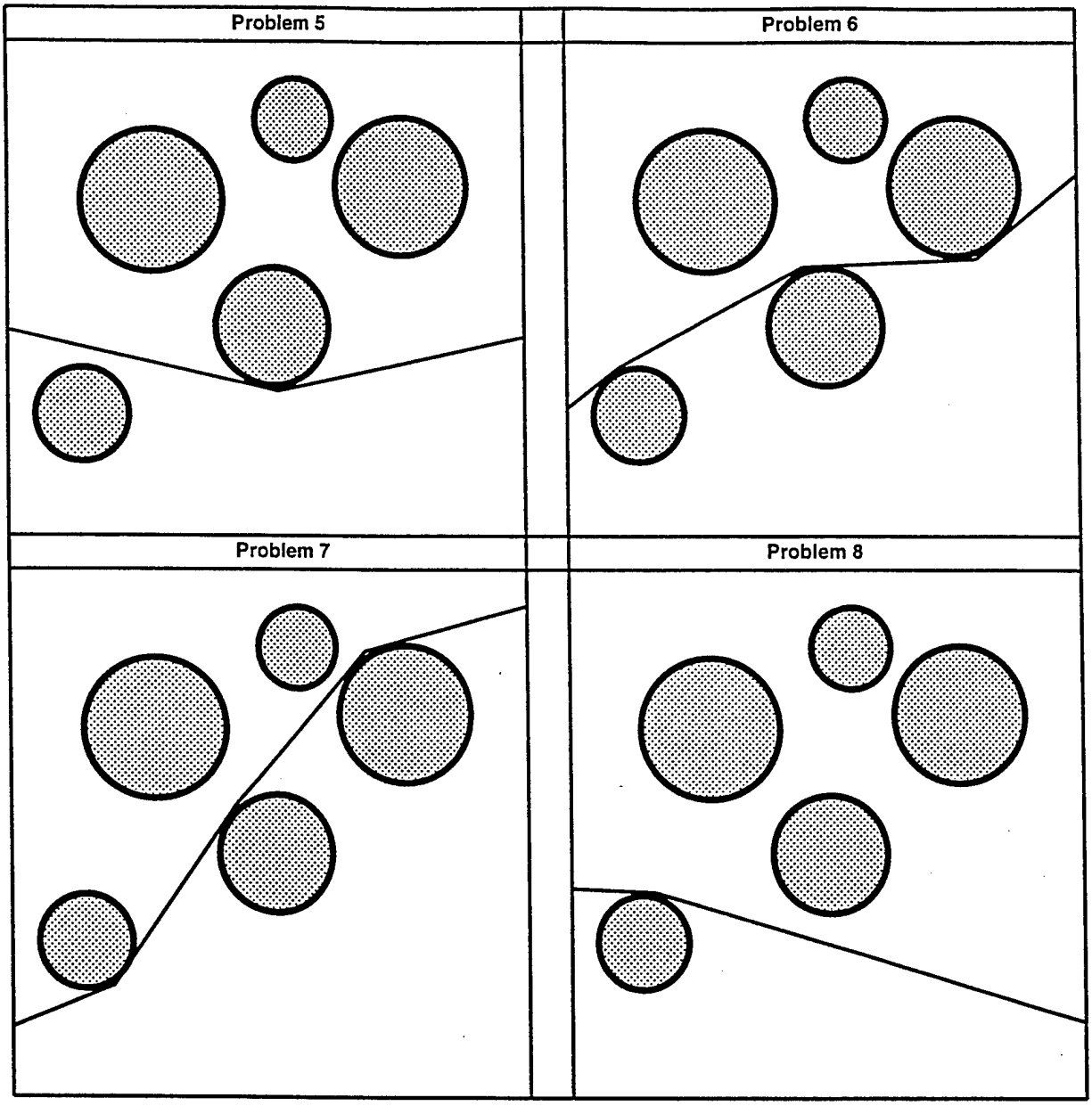

Figure 16: Display of Problems 5, 6, 7, and 8

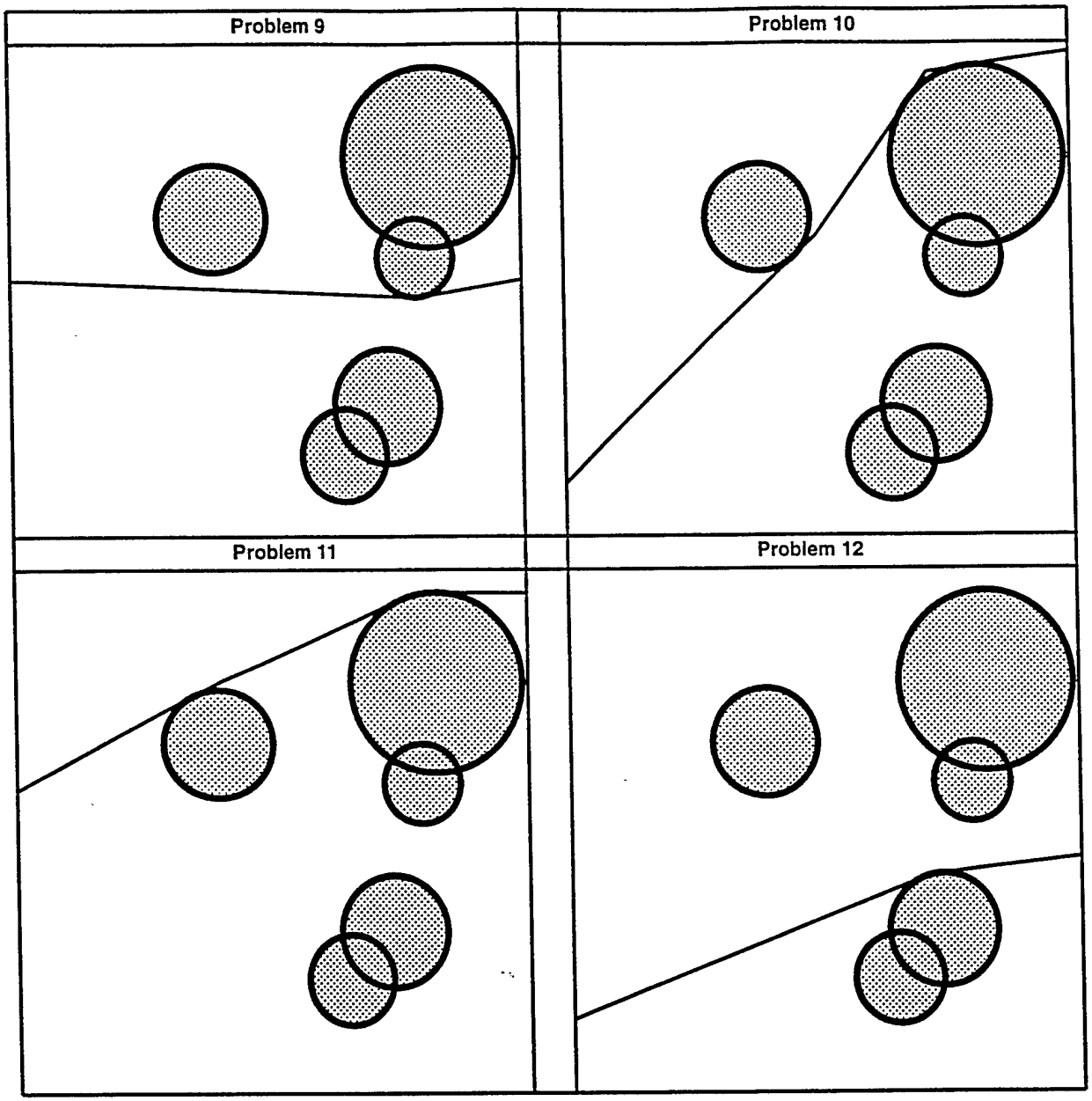

Figure 17: Display of Problems 9, 10, 11, and 12

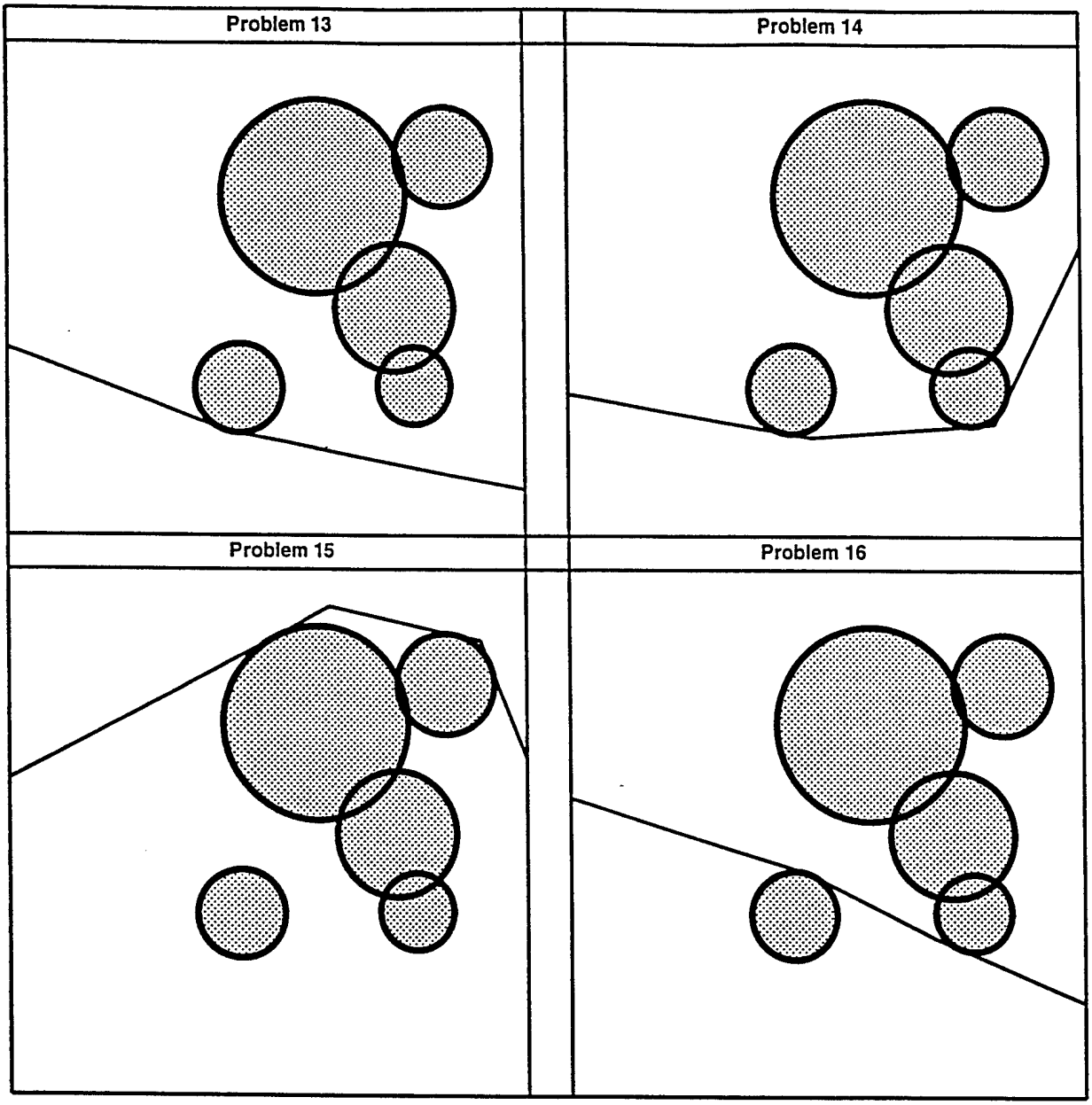

Figure 18: Display of Problems 13, 14, 15, and 16

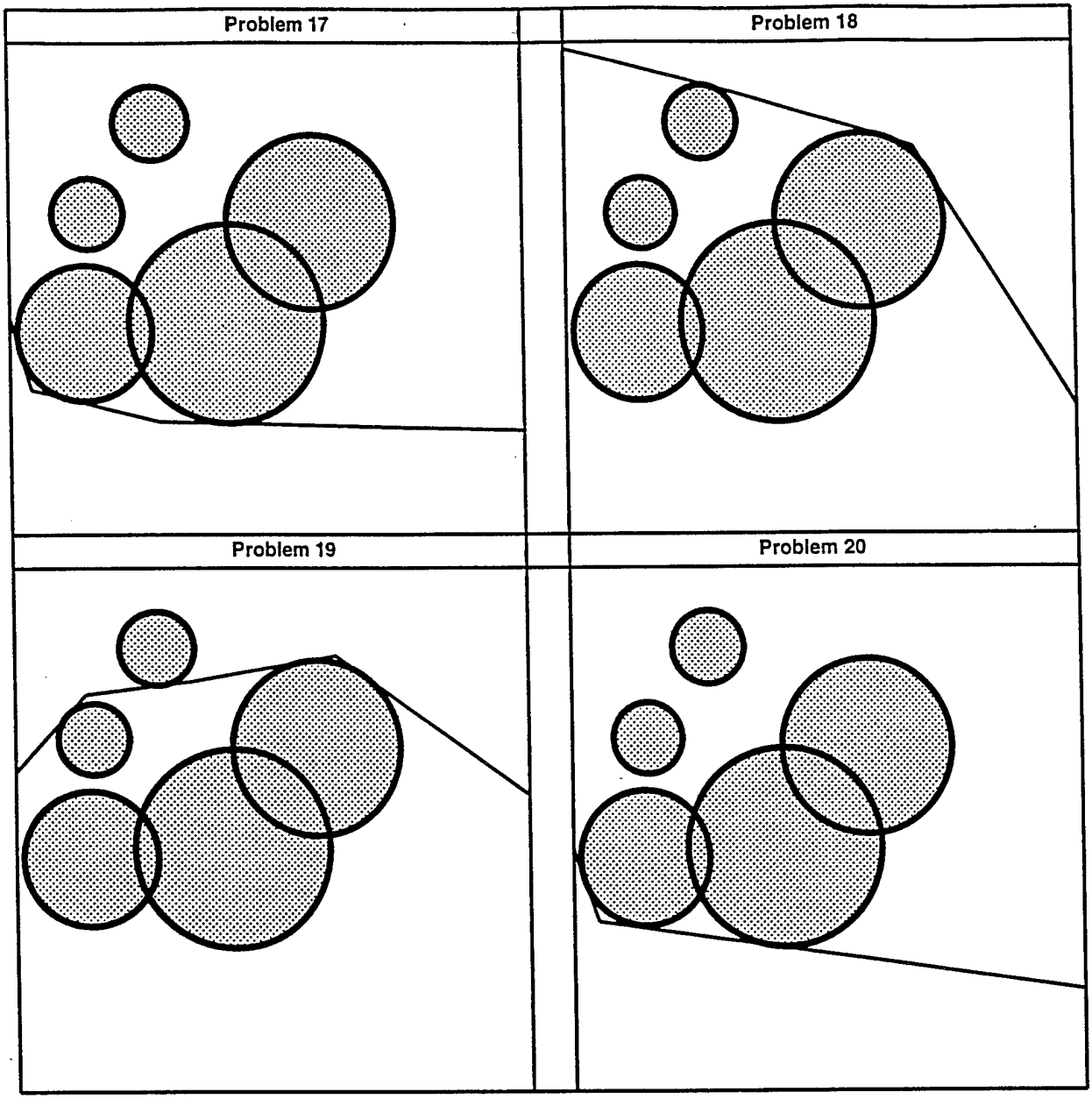

Figure 19: Display of Problems 17, 18, 19, and 20

Appendix C

Planning with a Probability Restriction

Technical Report 97-CSE-3

Cruise Missile Mission Planning with a Probability Side Constraint

by

R. V. Helgason
(helgason@seas.smu.edu)

J. L. Kennington
(jlk@seas.smu.edu)

K. H. Lewis
(klewis@seas.smu.edu)

Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275-0122

February 1997

Abstract

By representing threats as circles, the problem of mission planning for a cruise missile can be viewed as a computational geometry problem in which we seek a path composed of line segments from a launch site to a target site which skirts selected threats. In a previous investigation, we presented an effective algorithm of this type, called the circumscribed triangle algorithm. In this investigation, we extend the circumscribed triangle algorithm to incorporate a side constraint on the probability of a successful traversal. This allows the missile to take short cuts through threat areas, as long as the probability of success meets some user-specified criteria. In an empirical study on twenty randomly generated test problems using probabilities of 90%, 75%, and 60%, we found that the new algorithm works extremely well.

Acknowledgement

This research was supported in part by the Office of Naval Research under contract number N00014-96-1-0315.

1 INTRODUCTION

The cruise missile has recently become one of the Navy's most important weapon systems. During recent conflicts in the Persian Gulf, ship and submarine launched cruise missiles have been used extensively with great success. Targets which may be hundreds of miles inland can be attacked by these Navy weapons stationed in relatively safe positions offshore. In addition, cruise missiles can be used to disable SAM sites, making manned aircraft strikes substantially safer.

A mission for a cruise missile is defined as a path from a launch area to the target. The job of the mission planner is to determine the path that the missile will follow and program the missile with the appropriate waypoints and any TERCOM maps that will be used. A mission does not have to include TERCOM maps, but they can be used as part of the navigational package. Some missions involve only GPS waypoints.

Using current technology, missions are developed using a two-step process. In the first step, a mission planner uses a two-dimensional map to manually select a path from the launch site to the target. A software system enhances the path to include the vertical dimension and develops an estimate of the

probability that the mission will be successful. This process may be repeated several times until the mission planner is satisfied with the mission plan. Since a cruise missile costs approximately $650,000, the probability of success of a given mission is an important consideration for Navy management.

Several research groups have attempted to develop automatic methods to replace the manual part of the two-step approach. Most of these groups begin by placing a grid over the combat theatre and then applying a modification of Dijstra's [2] algorithm to obtain a path from the origin to the destination (see [1, 3, 5, 6, 8, 9]). Unfortunately, the grid can become very large and consequently the software can be fairly slow. In addition, the resulting path can involve numerous line segments.

In [4], the authors developed an effective algorithm for mission planning that does not involve the use of a grid. The threats are modelled as circles on a two-dimensional map, and the objective is to obtain a shortest path from a given origin to a given destination which does not pass through any threat circle. The circumscribed triangle algorithm constructs line segments by moving from a given segment point tangentially to the circumference of nearby threat regions, until a suitable path from origin to destination is found. In an empirical study, the algorithm always obtained a very good

mission plan.

The objective of this investigation is to incorporate a probability constraint into the circumscribed triangle algorithm. This allows for more realistic mission plans which fly through threat regions on their path to the target while maintaining a user-specified probability of success. The benefit of this extension is determined empirically on a test suite of twenty randomly-generated problems.

2 THE MISSION PLANNING PROBLEM

In this work, a threat is represented by a circle with a given center X and a given radius r. Let Z denote the point along the mission plan at which the missile is nearest X. If $||Z - X|| > r$, then the threat at X is too far away to harm the missile and the probability of a successful traversal is defined to be 1. Otherwise, some function $\mathrm{p}(Z, X, r)$ is used to determine the probability of a successful traversal. When there are multiple threats, the probability of a successful traversal is the product of the probabilities along each segment, taking into account all of the threats.

The problem of finding a feasible mission plan which incorporates a prob-

ability constraint, can be defined mathematically as follows:

Given a target probability P, two points M and $T \in R^2$, and K circles, with centers $X_1, ..., X_K \in R^2$ and radii $r_1, ..., r_K$, find a set of line segments $[Y_1, Y_2]$, $[Y_2, Y_3]$,..., $[Y_S, Y_{S+1}]$, with $Y_1 = M$ and $Y_{S+1} = T$, such that the probability of a successful traversal is at least P.

The quality of a mission plan is measured by the length of the path and the number of path segments. The best path is the straight line from M to T. If this is not feasible (i.e. the probability of a succcessful traversal is less than P), then we seek a short path having only a few segments.

3 THE ALGORITHM

The probabilistic incursion triangle algorithm allows the missile to fly through the interior of a threat region and employs a probability model to compute the probability of survival for a path from the missile location M to the target location T which consists of line segments which may either be tangent to a circular threat region or cut through the region (an incursion).

The algorithm operates within the same branch-and-bound algorithmic

structure used in the circumscribed triangle algorithm (see Section 3.2 of [4]). The major differences are (1) when attempting to construct a new segment by approaching a threat region from a previous segment endpoint, several equilateral triangles passing through the threat region in addition to the circumscribed triangle may be considered, and for each such triangle, potentially two points are added to the branch-and-bound queue, and (2) in searching for a final segment linking a previous segment endpoint to the target, that segment may intersect threat regions (with a consequent reduction in survival probability).

3.1 The Probability Model

For threat region k with center X_k and radius r_k, the probability model assumes that the probability of survival of a line segment approaching a threat region is a function only of the minimum distance from the segment to X_k. Further, the model assumes independence so that the overall survival probability for a path is the product of the survival probabilities for each segment in the path.

For developmental purposes we have hypothesized a crude probability

function which is a quadratic of the form

$$Pr(d) \;=\; ad^2 + c \;,$$

where for threat region k,

$$Pr(r_k) = .99 \text{ and } Pr(0) = .50 \;.$$

3.2 Incursion Triangles

Figure 6 illustrates a circumscribed triangle YiV_1V_2 and an incursion triangle $Y_iQ_1Q_2$ for a common threat region R_k. In the circumscribed triangle algorithm, with respect to the segments Y_iV_1 and Y_iV_2, the line segments T_1V_1 and T_2V_2 are searched first for a clear completion segment to the target, before the points V_1 and V_2 are added to the branch-and-bound queue. This also occurs in the probabilistic incursion triangle algorithm, but the completions can intersect threat regions if the total survival probability is at least P and the segments Y_iV_1 and Y_iV_2 will each have survival probability .99. Also, in the probabilistic incursion triangle algorithm, with respect to the segments Y_iQ_1 and Y_iQ_2, the line segments P_1Q_1 and P_2Q_2 are searched first for a completion to the target with total survival probability at least P, before the points V_1 and V_2 are added to the branch-and-bound queue.

The number of incursion triangles considered in approaching a threat region from a previous segment endpoint depends on the accumulated number of segments and an adjustable parameter $\hat{S}$ which is roughly interpreted as the expected number of segments in an optimal path. In the testing described in Section 4, $\hat{S}$ was set at 3. If S_i is the number of segments in a partial path from the missile location to segment endpoint Y_i and P_i is the survival probability of this partial path, incursion triangles with individual segment survival probabilities of

$$.99(P/P_i) \ , \ .99\left(\frac{P/P_i}{2}\right) \ , \ \ldots \ , \ .99\left(\frac{P/P_i}{\hat{S}-S}\right)$$

will be considered, unless these probabilities are very close to .99.

4 EMPIRICAL ANALYSIS

The twenty test problems presented in [4] were solved using $P = 90\%$, $P = 75\%$, and $P = 60\%$, and the results are summarized in Table 1. The deterministic algorithm is the circumscribed triangle algorithm described in [4]. Note that setting $P = 60\%$ subsantially reduces the number of way-points, but only reduces the total distance by 4%. The paths obtained by the deterministic algorithm and the new algorithm with $P = 60\%$ may be

found in Appendix A. For most problems, the two paths are similar, with the new algorithm taking short cuts through one or more threats. Problem 7 illustrates a case where the new algorithm selects a completely different path. For this problem, the deterministic algorithm produces a path in which all threats are above the path, while the new algorithm uses a mixed strategy. The deterministic strategy is unable to use a single straight line for any of these test problems, whereas the probabilistic algorithm finds this to be best for several of them. Based on these results, we believe that this approach could be used in the development of a grid-free auto-router, incorporating a probabilistic side constraint.

Table 1. Comparison of Probabilities for the Mission Planning Problem

Prob	Deterministic Algorithm		Probabilistic Algorithm					
			90%		75%		60%	
	Dist	Segmts	Dist	Segmts	Dist	Segmts	Dist	Segmts
1	957	3	955	3	952	3	949	3
2	853	3	849	3	842	3	834	3
3	853	4	851	3	843	3	803	3
4	945	3	939	3	925	3	905	3
5	795	2	794	2	789	2	783	2
6	878	4	876	4	868	4	857	4
7	1042	3	1040	3	1031	3	1012	3
8	805	2	805	2	804	1	804	1
9	782	2	782	2	780	2	780	1
10	1049	3	1044	3	1031	3	1012	3
11	847	3	845	3	838	3	835	2
12	820	2	819	2	817	2	815	1
13	810	2	809	2	807	2	807	1
14	945	3	942	3	861	3	809	2
15	966	3	962	3	950	3	780	2
16	837	3	837	3	835	3	835	1
17	865	3	857	3	848	3	825	3
18	1009	3	1005	3	994	3	977	3
19	900	4	895	3	882	3	857	3
20	862	2	851	3	837	2	820	3
Totals	17820	57	17757	56	17534	54	17099	47
Scaled	1.0	1.0	0.996	0.982	0.984	0.947	0.960	0.825

References

[1] A. Boroujerdi, C. Dong, Q. Ma, and B. Moret, "Joint Routing in Networks," undated technical report, Department of Computer Science, University of New Mexico, Albuquerque, NM.

[2] E. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik* 1 (1959) 269 - 271.

[3] R. Helgason, J. Kennington, and K. Lewis, "Finding Safe Paths in Networks," Technical Report 96-CSE-7, Department of Computer Science and Engineering, SMU, Dallas, TX 75275-0122 (1996).

[4] R. Helgason, A. Jayasuriya, J. Kennington, and K. Lewis, "Grid-Free Algorithms for the Two-Dimensional Mission Planning Problem for Cruise Missiles," Technical Report 96-CSE-10, Department of Computer Science and Engineering, SMU, Dallas, TX 75275-0122 (1996).

[5] J. Solka, J. Perry, B. Poellinger, and G. Rogers, "Fast Computation of Optimal Paths Using a Parallel Dijkstra Algorithm with Embedded Constraints," *Neurocomputing* 8, (1995) 195 - 212.

[6] J. Solka, J. Perry, B. Poellinger, and G. Rogers, "Autorouting Using a Parallel Dijkstra Algorithm with Embedded Constraints," undated technical report, The Naval Surface Warfare Center, Dahlgren, VA.

[7] B. Welch, *Practical Programming in Tcl and Tk*, Prentice-Hall, Inc., Upper Saddle River, NJ (1995).

[8] M. Zuniga and P. Gorman, "Threat Site Overflight Modeling for Strike Route Optimization," undated technical report, Naval Research Laboratory, Washington, D.C.

[9] M. Zuniga, J. Uhlmann, and J. Hofmann, "The Interdependent Joint Routing Problem: Description and Algorithmic Approach," undated technical report, Naval Research Laboratory, Washington, D.C.

APPENDIX A

SOLUTIONS

This appendix presents a graphical display of the twenty test problems along with the solutions obtained by the deterministic algorithm and the probabilistic algorithm. The black line is the solution obtained with a P set to 60% and the gray line is the mission obtained by the deterministic algorithm. The plots were obtained using Tcl and Tk Solver [7], which is part of our experimental computational package.

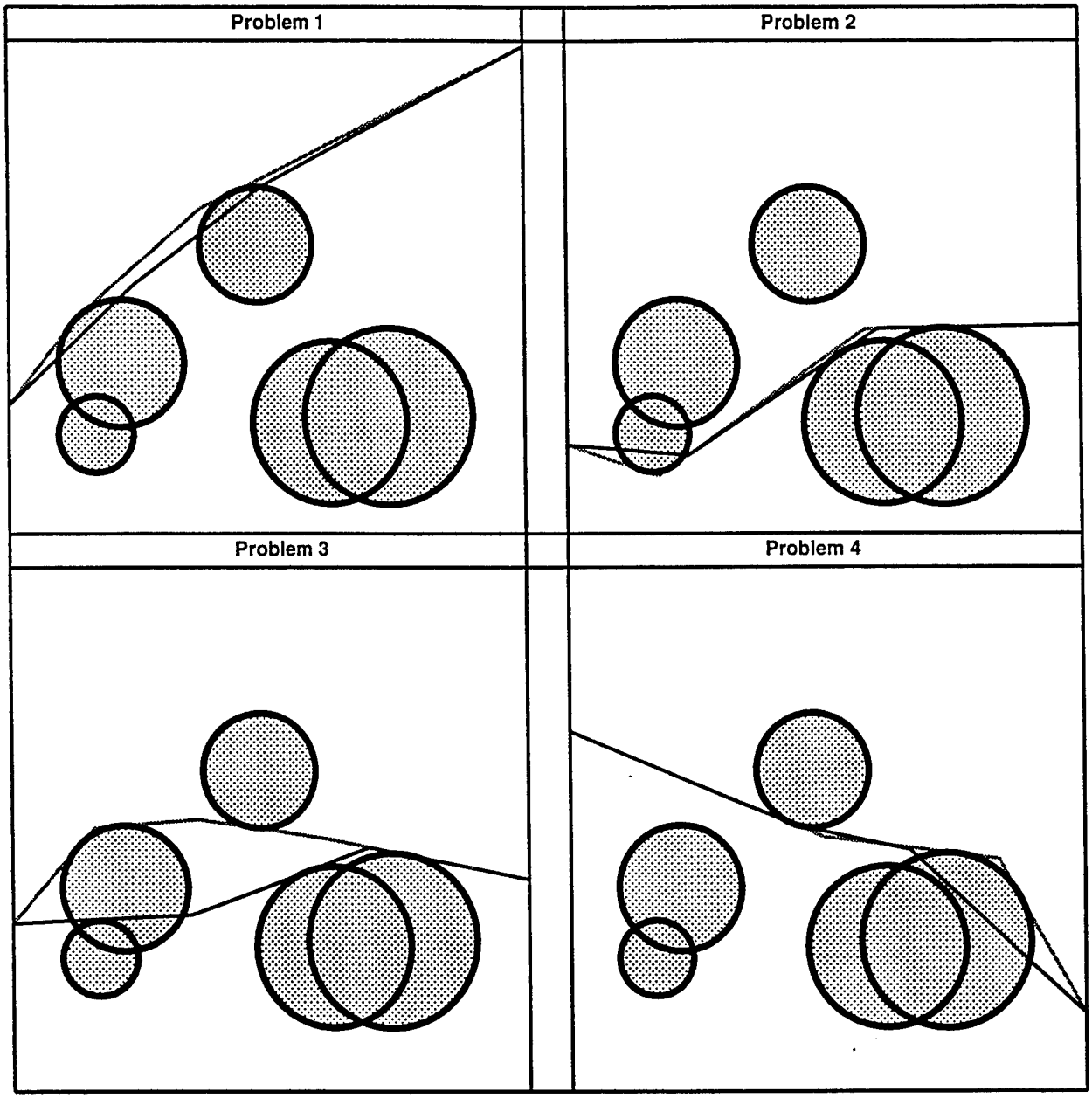

Figure 1: Display of Problems 1, 2, 3, and 4

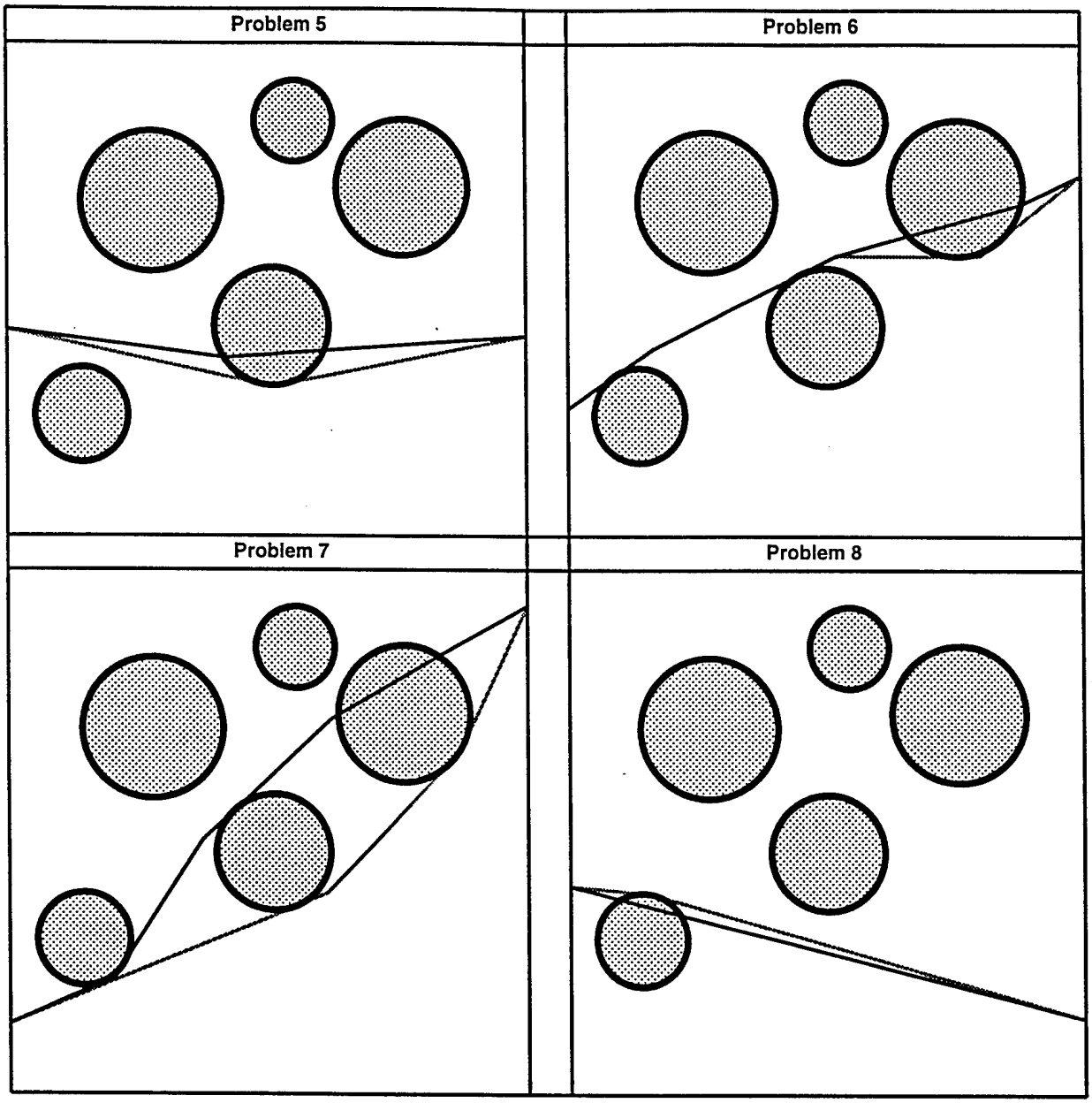

Figure 2: Display of Problems 5, 6, 7, and 8

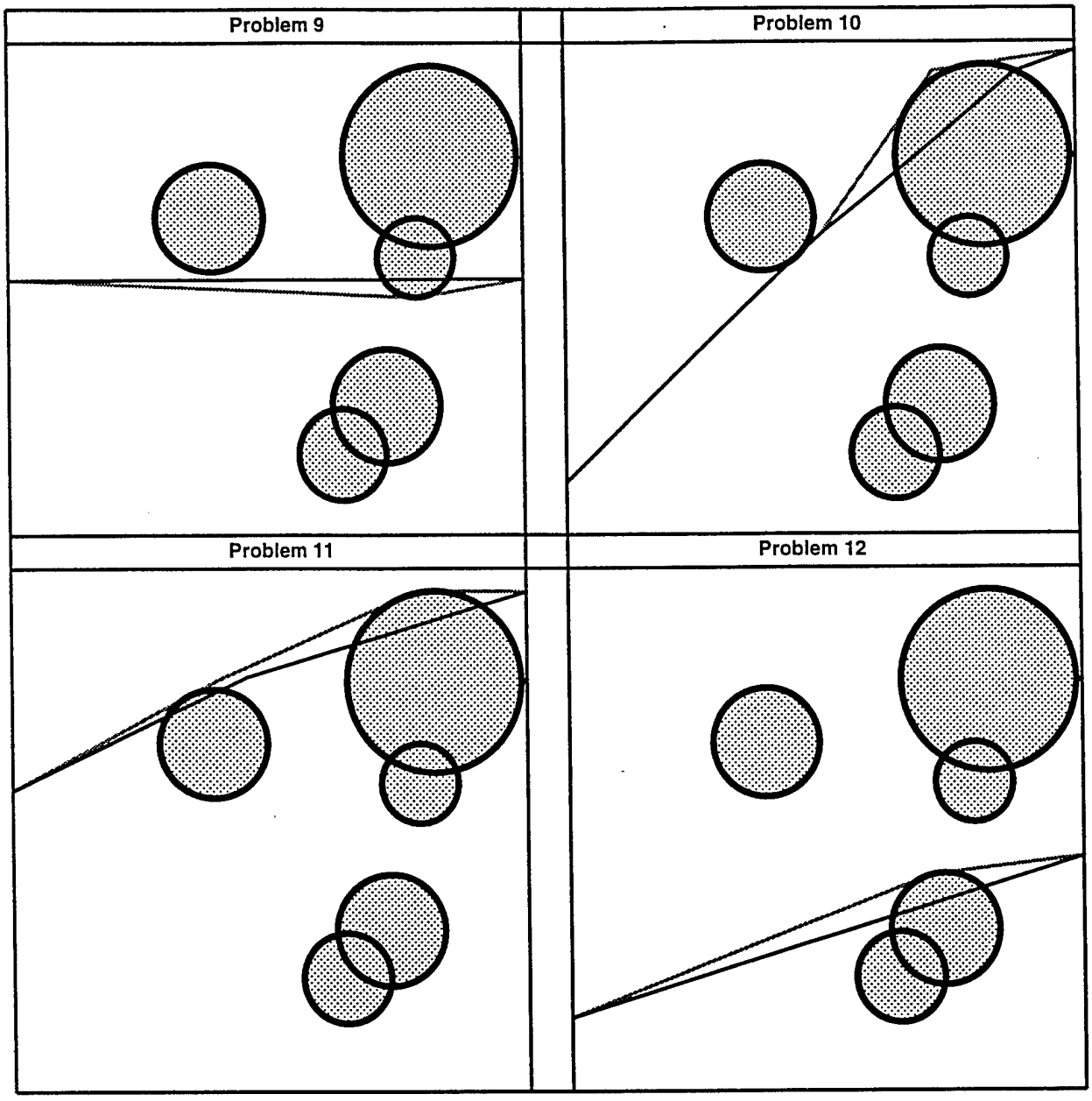

Figure 3: Display of Problems 9, 10, 11, and 12

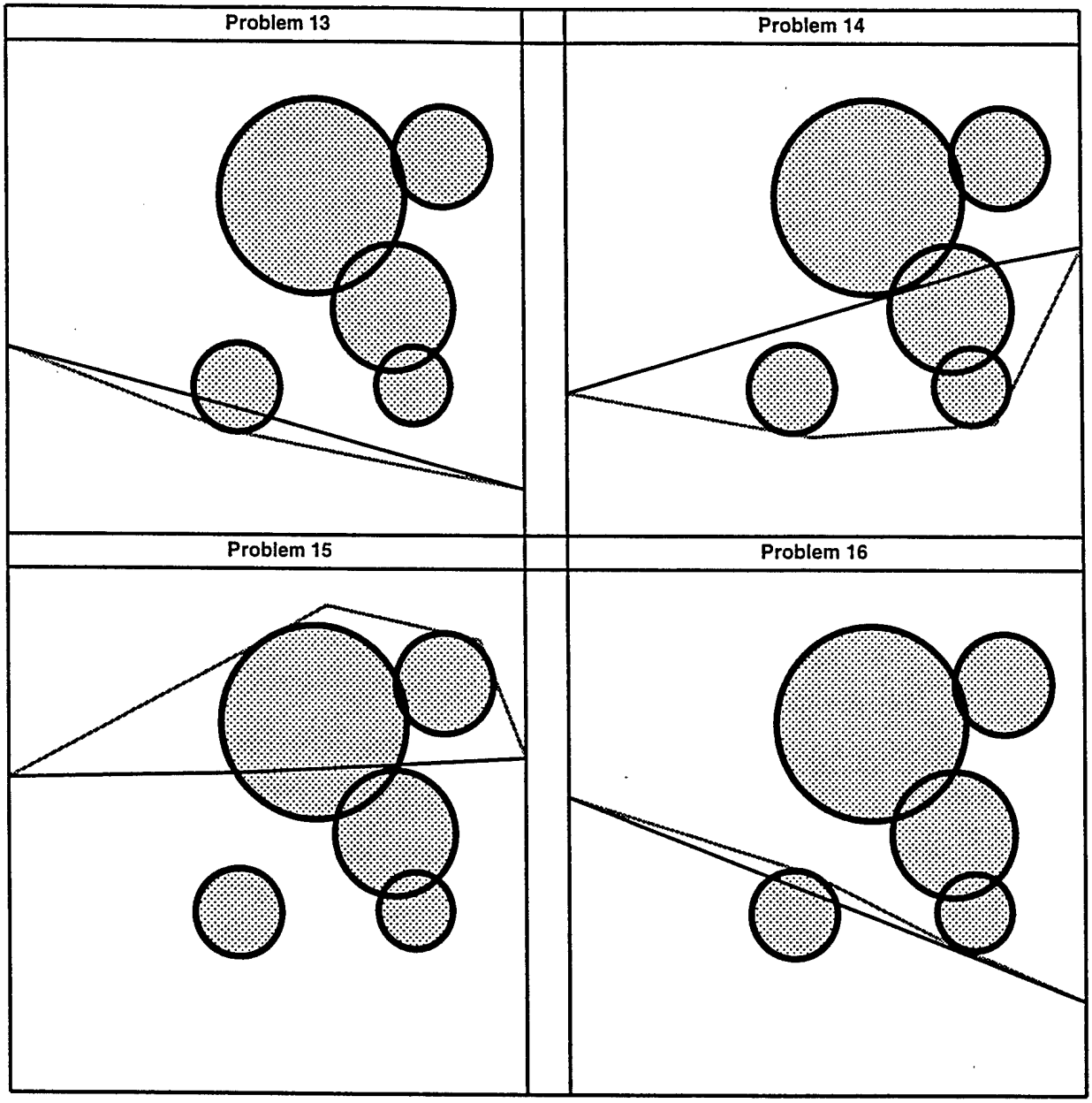

Figure 4: Display of Problems 13, 14, 15, and 16

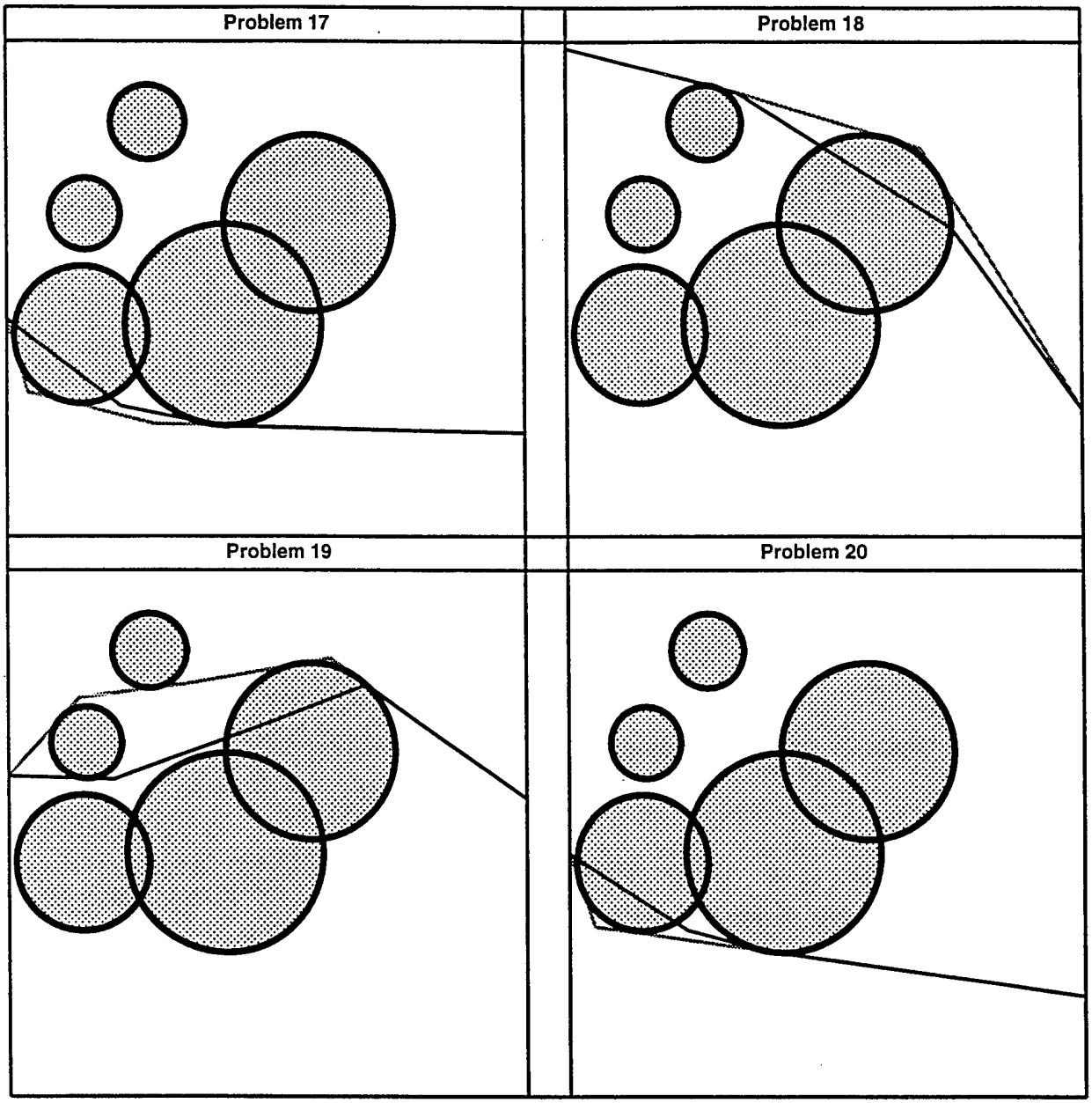

Figure 5: Display of Problems 17, 18, 19, and 20

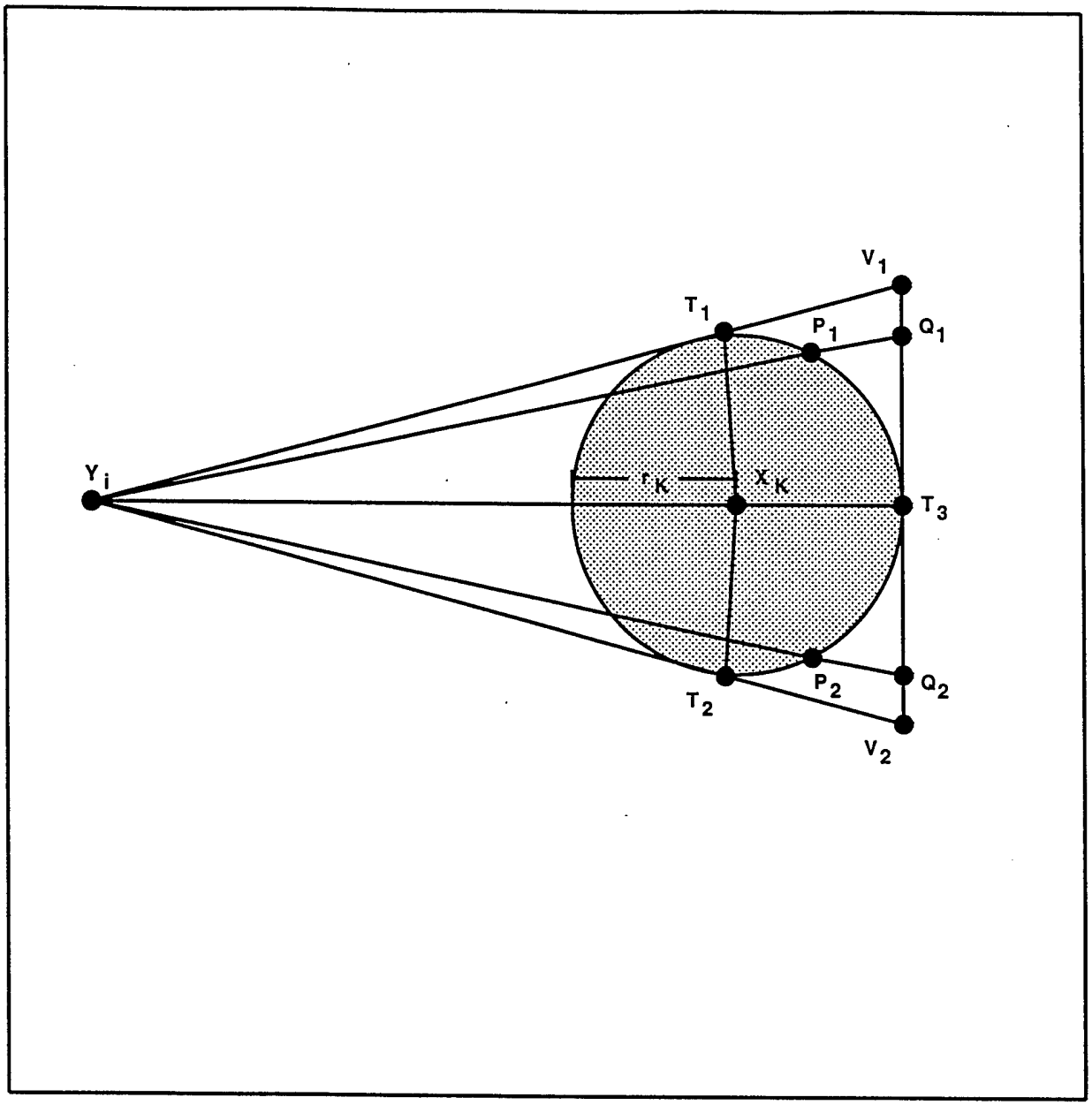

Figure 6: Circumscribing Triangle and Incursion Triangle with Vertex Y_i for Threat Region R_k

Appendix D

Distribution List

Donald Wagner ONR 311 Ballston Centre Tower One 800 North Quincy Street Arlington, VA 22217−5660	3 copies
Administrative Grants Officer Office of Naval Research Regional Office 4520 Executive Drive Suite 300 San Diego, CA 92121−3019	1 copy
Director, Naval Research Laboratory Attn: Code 2627 4555 Overlook Drive Washington, DC 20375−5326	1 copy
Defense Technical Information Center 8725 John J. Kingman Road SDTS 0944 Ft. Belvoir, VA 22060−6218	2 copies
Carol Voltmer Office of Research Administration SMU Dallas, TX 75275	1 copy